The Model Evolution Calculus with Equality

Peter Baumgartner
Programming Logics Group
Max-Planck-Institut für Informatik
baumgart@mpi-sb.mpg.de

Cesare Tinelli
Department of Computer Science
The University of Iowa
tinelli@cs.uiowa.edu

March 11, 2005, last modification on September 30, 2008

Abstract

In many theorem proving applications, a proper treatment of equational theories or equality is mandatory. In this paper we show how to integrate a modern treatment of equality in the Model Evolution calculus ($\mathcal{ME}$), a first-order version of the propositional DPLL procedure. The new calculus, $\mathcal{ME}_{\mathrm{E}}$, is a proper extension of the $\mathcal{ME}$ calculus without equality. Like $\mathcal{ME}$ it maintains an explicit *candidate model*, which is searched for by DPLL-style splitting. For equational reasoning $\mathcal{ME}_{\mathrm{E}}$ uses an adapted version of the ordered paramodulation inference rule, where equations used for paramodulation are drawn (only) from the candidate model. The calculus also features a generic, semantically justified simplification rule which covers many simplification techniques known from superposition-style theorem proving. Our main result is the correctness of the $\mathcal{ME}_{\mathrm{E}}$ calculus in the presence of very general redundancy elimination criteria.

1 Introduction

The Model Evolution ($\mathcal{ME}$) Calculus [BT03a] has recently been introduced by the authors of this paper as a first-order version of the propositional DPLL procedure [DLL62]. Compared to its predecessor, the FDPLL calculus [Bau00], it lifts to the first-order case not only the core of the DPLL procedure, the splitting rule, but also DPLL's simplification rules, which are crucial for effectiveness in practice.

Our implementation of the $\mathcal{ME}$ calculus, the Darwin system [BFT05], performs well in some domains, but, unsurprisingly, it generally performs poorly in domains with equality. In this paper we address this issue and propose an extension of the $\mathcal{ME}$ calculus with dedicated inference rules for equality reasoning. These rules are centered around a version the *ordered paramodulation* inference rule adapted to the $\mathcal{ME}$ calculus. The new calculus, $\mathcal{ME}_{\mathrm{E}}$, is a proper extension of the $\mathcal{ME}$ calculus without equality. Like $\mathcal{ME}$, it searches for a model of the input clause set by maintaining and incrementally modifying a finite representation, called a *context*, of a *candidate model* for the clause set. In $\mathcal{ME}_{\mathrm{E}}$, equations from the context, and only those, are used for ordered paramodulation

inferences into the current clause set. The used equations are kept together with the clause paramodulated into and act as passive constraints in the search for a model.

In this paper we present the calculus and prove its soundness and completeness. The completeness proof is obtained as an extension of the completeness proof of the $\mathcal{ME}$ calculus (without equality) by adapting techniques from the Bachmair/Ganzinger framework developed for proving the completeness of the superposition calculus [BG98, NR01, e.g.]. The underlying model construction technique allows us to justify a rather general simplification rule on semantic grounds. The simplification rule is based on a general redundancy criterion that covers many simplification techniques known from superposition-style theorem proving.

Related Work. Like $\mathcal{ME}$, the $\mathcal{ME}_{\mathrm{E}}$ calculus is related to *instance based methods (IMs)*, a family of calculi and proof procedures developed over the last ten years. What has been said in [BT03a] about $\mathcal{ME}$ in relation to IMs also applies to $\mathcal{ME}_{\mathrm{E}}$ when equality is not an issue, and the points made there will not be repeated here in detail. Instead, we focus on instance based methods that natively support equality reasoning.

Among them is Ordered Semantic Hyperlinking (OSHL) [PZ00]. OSHL uses rewriting and narrowing (paramodulation) with unit equations, but requires some other mechanism such as Brand's transformation to handle equations that appear in nonunit clauses.

To our knowledge there are only two instance-based methods that have been extended with dedicated equality inference rules for full equational clausal logic. One is called disconnection tableaux, which is a successor of the disconnection method [Bil96].[1] The other is the IM described in [GK03]. Both methods are conceptually rather different from $\mathcal{ME}$ in that the main derivation rules there are based on resolving *pairs* of complementary literals (connections) from *two* clauses, whereas $\mathcal{ME}$'s splitting rule is based on evaluating *all* literals of a *single* clause against a current candidate model.

The article [LS02] discusses various ways of integrating equality reasoning in disconnection tableaux. It includes a variant based on ordered paramodulation, where paramodulation inferences are determined by inspecting connections between literals of two clauses. Only comparably weak redundancy criteria are available.

The instance based method in [GK03] has been extended with equality in [GK04]. Beyond what has been said above there is one more conceptual difference, in that the inference step for equality reasoning is based on refuting, as a subtask, a set of unit clauses (which is obtained by picking clause literals).

Paper organization. We start with an informal explanation of the main ideas behind the $\mathcal{ME}_{\mathrm{E}}$ calculus in Section 2, followed by a more formal treatment of *contexts* and their associated interpretations in Section 3. Then, in Section 4, we present what we call *constrained clauses* and a way to perform equality reasoning on them. We describe the $\mathcal{ME}_{\mathrm{E}}$ calculus over constrained clauses in Section 6, and discuss its correctness in Section 7. Detailed proofs of all the results can be found in the appendix.

[1] Even in that early paper a paramodulation-like inference rule was considered, however a rather weak one.

2 Main Ideas

The $\mathcal{ME}$ calculus of [BT03a], and by extension the $\mathcal{ME}_{\mathrm{E}}$ calculus, is informally best described with an eye to the propositional DPLL procedure, of which $\mathcal{ME}$ is a first-order lifting. DPLL can be viewed as a procedure that searches the space of possible interpretations for a given clause set until it finds one that satisfies the clause set, if it exists. This can be done by keeping a current candidate model and *repairing* it as needed until it satisfies every input clause. The repairs are done incrementally by changing the truth value of one clause literal at a time, and involve a non-deterministic guess (a "split") on whether the value of a selected literal should be changed or kept as it is. The number of guesses is limited by a constraint propagation process ("unit propagation") that is able to deduce deterministically the value of some input literals.

Both $\mathcal{ME}$ and $\mathcal{ME}_{\mathrm{E}}$ lift this idea to first-order logic by maintaining a *first-order* candidate model, by identifying *instances* of input clauses that are falsified by the model, and by repairing the model incrementally until it satisfies all of these instances. The difference between the two calculi is that $\mathcal{ME}_{\mathrm{E}}$ works with *equational* models, or *E-interpretations*, that is, Herbrand interpretations in which the equality symbol is the only predicate symbols and always denotes a congruence relation.

The current E-interpretation is represented (or more precisely, induced) by a *context*, a finite set of non-ground equations and disequations directly processed by the calculus. Context literals can be built over two kinds of variables: *universal* and *parametric* variables. The difference between the two lies in how they constrain the possible additions of further literals to a context and, as a consequence, the possible repairs to its induced E-interpretation. As far as the induced E-interpretation is concerned, however, the two types of variables are interchangeable. The construction of this E-interpretation is best explained in two stages, each based on an ordering on terms/atoms: the usual instantiation preordering $\gtrsim$ with its strict subset $\gtrsim$, and an arbitrary reduction ordering $\succ$ total on ground terms. Using the first we associate to a context Λ, similarly to the $\mathcal{ME}$ calculus, a (non-equational) interpretation I_Λ. Roughly, and modulo symmetry of $\approx$, this interpretation satisfies a ground equation $s'' \approx t''$, over an underlying signature Σ, iff $s'' \approx t''$ is an instance of an equation $s \approx t$ in Λ without being an instance of any equation $s' \approx t'$ such that $s \approx t \gtrsim s' \approx t'$ and $s' \not\approx t' \in \Lambda$. For instance, if

$$\Lambda = \{f(u) \approx u, f(a) \not\approx a\}$$

where u is a (parametric) variable and the signature Σ consists of the unary function symbol f and the constant symbols a and b, then I_Λ is the symmetric closure of

$$\{f^{n+1}(b) \approx f^n(b) \mid n \geq 0\} \cup \{f^{n+1}(a) \approx f^n(a) \mid n \geq 1\}.$$

In general I_Λ is not an E-interpretation. Its purpose is merely to supply a set of candidate equations that determine the final E-interpretation induced by Λ. This E-interpretation, denoted by R^E_Λ, is defined as the smallest congruence on ground Σ-terms that includes a specific set R_Λ of ordered equations selected from I_Λ. The set

R_Λ is constructed inductively on the reduction ordering $\succ$ by adding to it an ordered equation $s \to t$ iff $s \approx t$ or $t \approx s$ is in I_Λ, $s \succ t$ and both s and t are irreducible wrt. the equations of R_Λ that are smaller than $s \to t$. This construction guarantees that R_Λ is a convergent rewrite system. In the example above, R_Λ is $\{f(b) \to b, f(f(a)) \to f(a)\}$ for any reduction ordering $\succ$; the E-interpretation R^E_Λ induced by Λ is the congruence closure of $\{f(b) \approx b, f(f(a)) \approx f(a)\}$. Since R_Λ is convergent by construction for any context Λ, any two ground Σ-terms are equal in R^E_Λ iff they have the same R_Λ-normal form.

Now that we have sketched how the E-interpretation is constructed, we can explain how the calculus detects the need to repair the current E-interpretation and how it goes about repairing it. To simplify the exposition we consider here only ground input clauses. A repair involves conceptually two steps: (i) determining whether a given clause C is false in the E-interpretation R^E_Λ, and (ii) if so, modifying Λ so that the new R^E_Λ satisfies it.

For step (i), by congruence it suffices to rewrite the literals of C with the rewrite rules R_Λ to normal form. If $C{\downarrow_{R_\Lambda}}$ denotes that normal form, then R^E_Λ fasifies C iff all equations in $C{\downarrow_{R_\Lambda}}$ are of the form $s \approx t$ with $s \neq t$, and all disequations are of the form $s \not\approx s$. In the earlier example, if

$$C = f(a) \approx a \vee f(f(a)) \approx b \vee f(b) \not\approx b$$

then

$$C{\downarrow_{R_\Lambda}} = f(a) \approx a \vee f(a) \approx b \vee b \not\approx b,$$

meaning that R^E_Λ indeed falsifies C.

For step (ii), we first point out that the actual repair needs to be carried out only on the literals of $C{\downarrow_{R_\Lambda}}$, not on the literals of C. More precisely, the calculus considers only the positive equations of $C{\downarrow_{R_\Lambda}}$, as the trivial disequations $s \not\approx s$ in it do not provide any usable information. To repair the E-interpretation it is enough to modify Λ so that R_Λ contains one of the positive equations $s \approx t$ of $C \downarrow_{R_\Lambda}$. Then, by congruence, R^E_Λ will also satisfy C, as desired. Concretely, Λ is modified by creating a choice point and adding to Λ one of the literals L of $C{\downarrow_{R_\Lambda}}$ or its complement. Adding L—which is possible only provided that neither L not its complement are contradictory, in a precise sense defined later, with Λ—-will make sure that the new R^E_Λ satisfies C. Adding the complement of L instead will not make C satisfiable in the new candidate E-model. However, it is necessary for soundness and marks some progress in the derivation because it will force the calculus to consider other literals of $C \downarrow_{R_\Lambda}$ for addition to the context.

Referring again to our running example, of the two positive literals of $C{\downarrow_{R_\Lambda}} = f(a) \approx a \vee f(a) \approx b \vee b \not\approx b$, only $f(a) \approx b$ can be added to the context $\Lambda = \{f(u) \approx u, f(a) \not\approx a\}$ because neither it nor its complement is contradictory with Λ (by contrast $f(a) \approx a$ is contradictory with Λ). With

$$\Lambda = \{f(u) \approx u, f(a) \not\approx a, f(a) \approx b\},$$

now

$$R_\Lambda = \{f(b) \to b, f(a) \to b\}$$

and $C{\downarrow_{R_\Lambda}}$ becomes $b \approx a \vee b \approx b \vee b \not\approx b$, which means that C is satisfied by R^E_Λ.

We point out that adding positive equations to the context is not always enough. Sometimes it is necessary to add negative equations, whose effect is to eliminate from R_Λ rewrite rules that cause the disequations of C to rewrite to trivial disequations. The calculus takes care of this possibility as well. To achieve that we found it convenient to have $\mathcal{ME}_{\mathrm{E}}$ work with a slightly generalized data structure. More precisely, instead of clauses C we consider *constrained clauses* $C \cdot \Gamma$, where Γ is a set of rewrite rules. The constraint Γ consists just of those (instances of) *equations* from a context Λ that were used to obtain C from some input clause (whose constraint is empty).

Reusing our example, the clause C would be represented as the constraint clause

$$C \cdot \Gamma = f(a) \approx a \vee f(f(a)) \approx b \vee f(b) \not\approx b \cdot \emptyset,$$

with its R_Λ-normal form being

$$C \downarrow_{R_\Lambda} \cdot \Gamma = f(a) \approx a \vee f(a) \approx b \vee b \not\approx b \cdot f(f(a) \to f(a), f(b) \to b$$

for $\Lambda = \{f(u) \approx u, f(a) \not\approx a\}$. Now, the rewrite rule $f(b) \to b$ used to obtain the normal form is available in the constraint part, as written. The calculus may add its negation $f(b) \not\approx b$ to Λ, with the effect of removing $f(b) \to b$ from R_Λ. The resulting context and rewrite system would be, respectively,

$$\Lambda'' = \{f(u) \approx u, f(a) \not\approx a, f(b) \not\approx b\},$$

and

$$R_{\Lambda''} = \{f(f(b)) \to f(b), f(f(a)) \to f(a)\}.$$

It is easy to see that the new $I^E_{R_\Lambda}$ satisfies C as well, as desired.

While the above informal description illustrates the main ideas behind $\mathcal{ME}_{\mathrm{E}}$, it is not entirely faithful to the actual calculus as defined later in the paper. Perhaps the most significant differences to mention here are that (i) the calculus works with non-ground clauses as well (by treating them, as usual in refutation-based calculi, as schematic for their ground instances and relying heavily on unification), and (ii) the normal form of a constrained clause is not derived in one sweep, as presented above. Instead the calculus, when equipped with a fair strategy, derives all intermediate constrained clauses as well. It does so by a suitably defined paramodulation rule, where the equations paramodulating (only) into the clause part of a constrained clause are drawn from the current context Λ. The rationale is that the rewrite system R_Λ is in general not available to the calculus. Hence rewriting (ground) clause literals with rules from R_Λ, which would theoretically suffice to obtain a complete calculus at the ground level, is approximated by ordered paramodulation with equations from Λ instead.

3 Contexts and Induced Interpretations

We start with some formal preliminaries. Most of the notions and notation we use in this paper are the standard ones in the field. We report here only notable differences and additions.

We will use two disjoint, infinite sets of variables: a set X of *universal* variables, which we will refer to just as variables, and another set V, which we will always refer to as *parameters*. We will use u and v to denote elements of V and x and y to denote elements of X. We fix a signature Σ throughout the paper and denote by Σ^{sko} the expansion of Σ obtained by adding to Σ an infinite number of fresh (Skolem) constants. Unless otherwise specified, when we say term we will mean Σ^{sko}-term. If t is a term we denote by $\mathcal{V}ar(t)$ the set of t's variables and by $\mathcal{P}ar(t)$ the set of t's parameters. A term t is *ground* iff $\mathcal{V}ar(t) = \mathcal{P}ar(t) = \emptyset$.

A substitution ρ is a *renaming on* $W \subseteq (V \cup X)$ iff its restriction to W is a bijection of W onto itself; ρ is simply a *renaming* if it is a renaming on $V \cup X$. A substitution σ is *p-preserving* (short for parameter preserving) if it is a renaming on V. If s and t are two terms, we write $s \gtrsim t$, iff there is a substitution σ 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 ρ such that $s\rho = t$. We write $s \gtrsim\!\!\!\!\!\!\not\;\; t$ if $s \gtrsim t$ but $s \not\sim t$. We write $s \geq t$ and say that t is *a p-instance of* s iff there is a p-preserving substitution σ such that $s\sigma = t$. We say that s is *a p-variant of* t, and write $s \simeq t$, iff $s \geq t$ and $t \geq s$; equivalently, iff there is a p-preserving renaming ρ such that $s\rho = t$. The notation $s[t]_p$ means that the term t occurs in the term s at position p, as usual.

All of the above is extended from terms to literals in the obvious way.

In this paper we restrict to equational clause logic. Any atom A that is originally not an equation can be represented as the equation $A \approx \mathbf{t}$, where $\mathbf{t}$ is some distinguished constant not appearing elsewhere. This move is harmless, in particular from an operational point of view (as will become clear from the design of the calculus) if one works with two-sorted signatures. Therefore, and essentially without loss of generality, we assume that the only predicate symbol in Σ is $\approx$. An atom then is always an equation, and a literal then is always an equation or the negation of an equation. Literals of the latter kind, i.e., literals of the form $\neg(s \approx t)$ are also called *negative equations* and generally written $s \not\approx t$ instead. We call a literal *trivial* if it is of the form $t \approx t$ or $t \not\approx t$. We denote literals by the letters K and L. We denote by $\overline{L}$ the complement of a literal L, and by L^{sko} the result of replacing each variable of L by a fresh Skolem constant in $\Sigma^{\text{sko}} \setminus \Sigma$. We denote clauses by the letters C and D, and the empty clause by $\Box$. We will write $L \vee C$ to denote a clause obtained as the disjunction of a (possibly empty) clause C and a literal L.

A *(Herbrand) interpretation* I is a set of ground Σ-equations—those that are true in the interpretation. Satisfiability/validity of ground Σ-literals, Σ-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 Σ-literal or a Σ-clause (set). Since every interpretation defines in effect a binary relation on ground Σ-terms, and every binary relation on such terms defines an interpretation, we will identity the two notions in the following. An *E-interpretation* is an interpretation that is also a congruence relation on the Σ-terms. If I is an interpretation, we denote by I^{E} the smallest congruence relation

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

on the Σ-terms that includes I, which is an E-interpretation. We say that I *E-satisfies* F iff $I^{\mathrm{E}} \models F$. Instead of $I^{\mathrm{E}} \models F$ we generally write $I \models_{\mathrm{E}} F$. We say that F *E-entails* F', written $F \models_{\mathrm{E}} F'$, iff every E-interpretation that satisfies F also satisfies F'. We say that F and F' are *E-equivalent* iff $F \models_{\mathrm{E}} F'$ and $F' \models_{\mathrm{E}} F$.

The Model Evolution calculus, with and without equality, works with sequents of the form $\Lambda \vdash \Phi$, where Λ is a finite set of literals possibly with variables or with parameters called a context, and Φ is a finite set of clauses possibly with variables. As in [BT03a], we impose for simplicity that literals in a context can contain parameters or variables but not both, but this limitation can be overcome. The defining feature of the calculus, modeled after FDPLL, is the rôle contexts play in driving the derivation and the model generation process.

The following definitions are taken from [BT03a], except for one minor change, which will be explained below.

Definition 3.1 (Context [BT03a])
A *context* is a set of the form $\{\neg v\} \cup S$ where $v \in V$ and S is a finite set of literals each of which is parameter-free or variable-free. □

Differently from [BT03a], we implicitly treat any context Λ as if it contained the symmetric version of each of its literals. For instance, if $\Lambda = \{\neg v, f(u) \approx a, f(x) \not\approx x\}$ then $a \approx f(u), f(u) \approx a, x \not\approx f(x), f(x) \not\approx x$ are all considered to be literals of Λ, and we write, for instance, $a \approx f(u) \in \Lambda$.

Where L is a literal and Λ a context, we write $L \in_{\sim} \Lambda$ if L is a variant of a literal in Λ, write $L \in_{\simeq} \Lambda$ if L is a p-variant of a literal in Λ, and write $L \in_{\geq} \Lambda$ if L is a p-instance of a literal in Λ.

A literal L is *contradictory with* a context Λ iff $L\sigma = \overline{K}\sigma$ for some $K \in_{\simeq} \Lambda$ and some p-preserving substitution σ. A context Λ is *contradictory* iff it contains a literal that is contradictory with Λ. Referring to the context Λ above, $f(v) \not\approx a$, $a \not\approx f(v), a \approx f(a), f(a) \approx a$ all are contradictory with Λ. Notice that an equation $s \approx t$ is contradictory with a context Λ if and only if $t \approx s$ is so. The same applies to negative equations.

We will work only with non-contradictory contexts. Thanks to the next two notions, such contexts can be used as finite denotations of (certain) Herbrand interpretations. Let L be a literal and Λ a context. A literal K is *a most specific generalization (msg) of* L *in* Λ iff $K \gtrsim L$ and there is no $K' \in \Lambda$ such that $K \gtrsim K' \gtrsim L$.

Definition 3.2 (Productivity [BT03a])
Let L be a literal, C a clause, and Λ a context. A literal K *produces* L *in* Λ iff (i) K is an msg of L in Λ, and (ii) there is no $K' \in_{\geq} \Lambda$ such that $K \gtrsim \overline{K'} \gtrsim L$. The context Λ *produces* L iff it contains a literal K that produces L in Λ. □

Notice that a literal K produces a literal L in a context Λ if and only if K produces the symmetric version of L in Λ. For instance, the context Λ above produces $f(b) \approx a$

and $a \approx f(b)$ but Λ produces neither $f(a) \approx a$ nor $a \approx f(a)$. Instead it produces both $a \not\approx f(a)$ and $f(a) \not\approx a$.

A non-contradictory context Λ uniquely induces a (Herbrand) Σ-interpretation I_Λ, defined as follows:

$$I_\Lambda := \{l \approx r \mid l \approx r \text{ is a positive ground } \Sigma\text{-equation and } \Lambda \text{ produces } l \approx r\}$$

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

A consequence of the presence of the pseudo-literal $\neg v$ in every context Λ is that Λ produces L or $\overline{L}$ for every literal L. Moreover, it can be easily shown that whenever $I_\Lambda \models L$ then Λ produces L, even when L is a negative literal. This fact provides a "syntactic" handle on literals satisfied by I_Λ. The induced interpretation I_Λ is not an E-interpretation in general.[3] But we will use it to define a unique E-interpretation associated to Λ.

4 Equality Reasoning on Constrained Clauses

The $\mathcal{ME}_\mathrm{E}$ calculus operates with *constrained clauses*, defined below. In this section we will introduce derivation rules for equality reasoning on constrained clauses. These derivation rules will be used by the $\mathcal{ME}_\mathrm{E}$ calculus in a modular way. The section concludes with a first soundness and completeness result, which will serve as a lemma for the completeness proof of the $\mathcal{ME}_\mathrm{E}$ calculus.

As an important preliminary remark, whenever the choice of the signature makes a difference in this section, e.g. in the definition of grounding substitution, we always implicitly meant the signature Σ, not the signature Σ^{sko}.

4.1 Constrained Clauses

A *(rewrite) rule* is an expression of the form $l \to r$ where l and r are Σ-terms. Given a parameter-free Σ-clause $C = L_1 \vee \cdots \vee L_n$ and a set of parameter-free Σ-rewrite rules $\Gamma = \{A_1, \ldots, A_m\}$, the expression $C \cdot \Gamma$ is called a *constrained clause (with constraint Γ)*. Instead of $C \cdot \{A_1, \ldots, A_m\}$ we generally write $C \cdot A_1, \ldots, A_m$. The notation $C \cdot \Gamma, A$ means $C \cdot \Gamma \cup \{A\}$.

A constrained clause $C \cdot \Gamma$ is a *constrained clause without expansion constraints* iff Γ contains no *expansion rules*, i.e., rules of the form $x \to t$, where x is a variable and t is a term. A *constrained clause set without expansion constraints* is a constrained clause set that consists of constrained clauses without expansion constraints. The $\mathcal{ME}_\mathrm{E}$ calculus works only with such constrained clause sets.[4] As a notational convention, we general use the letter Φ to denote sets of constrained clauses, and the letter Ψ to denote sets of ordinary clauses.

[3] In fact, in the earlier example $a \approx f(f(a)) \notin I_\Lambda$.

[4] As will become clear later, disallowing expansion constraints comes from the fact that paramodulation into variables is unnecessary in $\mathcal{ME}_\mathrm{E}$ as well.

Applying a substitution σ to $C \cdot \Gamma$, written as $(C \cdot \Gamma)\sigma$, means to apply σ to C and all rewrite rules in Γ. A constrained clause $C \cdot \Gamma$ is *ground* iff both C and Γ are ground. If γ is a substitution such that $(C \cdot \Gamma)\gamma$ is ground, then $(C \cdot \Gamma)\gamma$ is called a *ground instance* of $C \cdot \Gamma$, and γ is called a *grounding substitution* for $C \cdot \Gamma$. We say that $C \cdot \Gamma$ *properly subsumes* $C' \cdot \Gamma'$ iff there is a substitution σ such that $C\sigma \subset C'$ and $\Gamma\sigma \subseteq \Gamma'$ or $C\sigma \subseteq C'$ and $\Gamma\sigma \subset \Gamma'$. We say that $C \cdot \Gamma$ *non-properly subsumes* $C' \cdot \Gamma'$ iff there is a substitution σ such that $C\sigma = C'$ and $\Gamma\sigma = \Gamma'$. The constrained clauses $C \cdot \Gamma$ and $C' \cdot \Gamma'$ are *variants* iff $C \cdot \Gamma$ non-properly subsumes $C' \cdot \Gamma'$ and vice versa.[5] For a set of constrained clauses Φ, Φ^{gr} denotes the set of all ground Σ-instances of all constrained clauses in Φ.

In principle, a constraint clause $C \cdot \Gamma = L_1 \vee \cdots \vee L_m \cdot l_{m+1} \to r_{m+1}, \ldots, l_n \to r_n$ could be understood as standing for the ordinary clause $L_1 \vee \cdots \vee L_m \vee l_{m+1} \not\approx r_{m+1} \vee \cdots \vee l_n \not\approx r_n$, which we call the *clausal form of* $C \cdot \Gamma$ and denote by $(C \cdot \Gamma)^c$. In effect, however, constrained clauses and their clausal forms are rather different from an operational point of view. The derivation rules for equality reasoning below, in particular paramodulation, are *never* applied to constraints—as a consequence, the calculus cannot be said to be a resolution calculus.

4.2 Orderings and Derivation Rules

We suppose as given a reduction ordering $\succ$ that is total on ground Σ-terms. [6] The non-strict ordering induced by $\succ$ is denoted by $\succeq$, and $\prec$ and $\preceq$ denote the converse of $\succ$ and $\succeq$.

The reduction ordering $\succ$ has to be extended to rewrite rules, equations and constrained clauses. Following usual techniques [BG98, NR01, e.g.], rewrite rules and equations are compared by comparing the multisets of their top-level terms with the multiset extension of the base ordering $\succ$. There is no need in our framework to distinguish between positive and negative equations. It is important, though, that when comparing constrained clauses the clause part is given precedence over the constraint part. This can be achieved by defining $C \cdot \Gamma \succ C' \cdot \Gamma'$ iff (C, Γ) is strictly greater than (C', Γ') in the lexicographical ordering over the multiset extension of the above ordering on equations and rewrite rules. The following makes this precise.

Equations are compared by comparing the multisets of their top-level terms. More formally, let $M^l(s \approx t) = M^l(s \not\approx t) := \{s, t\}$ and define an ordering on literals $\succ^l$ as $L_1 \succ^l L_2$ iff $M^l(L_1) \succ_m M^l(L_1)$, where $\succ^m$ is the extension of $\succ$ to multisets. Rewrite rules are also compared with this ordering.

The calculus has to compare constrained clauses. To define the ordering $\succ^c$ on

[5] The reason to distinguish between proper and non-proper subsumption is that proper subsumption is covered by a more general notion of redundancy, while non-proper subsumption is not. Notice that the general redundancy criterion in the superposition calculus have the same limitation. Fortunately, non-proper subsumption tests can be used as well to eliminate clauses.

[6] 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 δ.

constrained clauses, associate to a constrained clause $C \cdot \Gamma$ the multiset

$$M^c(C \cdot \Gamma) = \bigcup_{L \in C} (M^l(L) \cup M^l(L)) \cup \bigcup_{A \in \Gamma} M^l(A) \ .$$

In words, M^c collects from $C \cdot \Gamma$ all the top-level terms from all equations in C and in Γ, but takes the terms from C twice and ignores whether the equations are positive or negative. This way, higher weight is given to the equations in C than those in Γ. Now define $C_1 \cdot \Gamma_1 \succ^c C_2 \cdot \Gamma_2$ iff $M(C_1 \cdot \Gamma_1) \succ^m M(C_2 \cdot \Gamma_2)$. In the sequel we will use the symbol $\succ$ instead of $\succ^c$.

For instance, if $a \succ b$ then

$$a \not\approx b \cdot \emptyset \succ b \not\approx b \cdot a \to b \succ \Box \cdot a \to b,$$

as $\{a, b, a, b\} \succ^m \{b, b, b, b, a, b\} \succ^m \{a, b\}$. This example helps to explain the motivation of the ordering: the derivation rules below must work in an order-decreasing way. For instance, the Para rule below allows to derive from the constrained clause $a \not\approx b \cdot \emptyset$ and the equation $a \approx b$ the new constrained clause $b \not\approx b \cdot a \to b$, and the Ref rule allows to derive $\Box \cdot a \to b$ then. Notice these steps indeed derive smaller constrained clauses.

Notice also that in superposition-type calculi usually a different ordering is used, which gives negative equations higher weight than positive equations (on the same terms). Such an ordering would work for us as well.

Derivation Rules. We first define two auxiliary derivation rules for equality reasoning on constrained clauses. The rules will be used later in the $\mathcal{ME}_{\mathrm{E}}$ calculus.

$$\mathsf{Ref}(\sigma) \quad \frac{s \not\approx t \vee C \cdot \Gamma}{(C \cdot \Gamma)\sigma} \quad \text{if } \sigma \text{ is a mgu of } s \text{ and } t.$$

We write $s \not\approx t \vee C \cdot \Gamma \Rightarrow_{\mathsf{Ref}(\sigma)} (C \cdot \Gamma)\sigma$ to denote a Ref inference.[7]

$$\mathsf{Para}(l \approx r, \sigma) \quad \frac{L[t]_p \vee C \cdot \Gamma}{(L[r]_p \vee C \cdot \Gamma, l \to r)\sigma} \quad \text{if} \begin{cases} t \text{ is not a variable,} \\ \sigma \text{ is a mgu of } t \text{ and } l, \text{ and} \\ l\sigma \not\preceq r\sigma. \end{cases}$$

We write $L[t]_p \vee C \cdot \Gamma \Rightarrow_{\mathsf{Para}(l \approx r, \sigma)} (L[r]_p \vee C \cdot \Gamma, l \to r)\sigma$ to denote a Para inference.

Ground Inferences. Suppose $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$, for some constrained clause $C' \cdot \Gamma'$, where D stands for $\mathsf{Ref}(\sigma)$ or $\mathsf{Para}(l \approx r, \sigma)$. If $C \cdot \Gamma$ is ground, and also $l \approx r$ in the case of Para is ground, then $C' \cdot \Gamma'$ is ground as well and σ is the empty substitution ϵ. The inference is called a *ground inference* then.

Let γ be a grounding substitution for $C \cdot \Gamma$ and for $l \approx r$. If $\gamma = \sigma\delta$ for some substitution δ, then $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ might hold or not, where D' stands for $\mathsf{Ref}(\epsilon)$ or $\mathsf{Para}((l \approx r)\gamma, \epsilon)$. If it holds, the inference is called a *ground instance via* γ of $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ then.

[7] An *inference* is an instance of a derivation rule that satisfies the rule's side condition.

Observe that a ground inference via γ of any Ref inference exists, whenever $\gamma = \sigma\delta$, for some some substitution δ. However, in the case of Para, not every grounding substitution $\gamma = \sigma\delta$, for some some substitution δ, yields a ground inference $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{Para}((l\approx r)\gamma,\epsilon)} (C' \cdot \Gamma')\delta$. It is not a ground inference precisely when $l\gamma \not\preceq r\gamma$ is not satisfied.

Also, if $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{Para}((l\approx r)\gamma,\epsilon)} (C' \cdot \Gamma')\delta$ is a ground inference, its "lifted" version $C \cdot \Gamma \Rightarrow_{\mathsf{Para}(l\approx r,\sigma)} C' \cdot \Gamma'$ need not exist, for any substitution σ. The reason is the occurence p paramodulated into on the ground need not exist on the first-order level or could be a variable.

As in the superposition calculus, *model construction*, *redundancy* and *saturation* are core concepts for the understanding of the $\mathcal{ME}_{\mathrm{E}}$ calculus.

4.3 Model Construction

A rewrite system is a set of Σ-rewrite rules. A ground rewrite system R is *ordered by* $\succ$ iff $l \succ r$, for every rule $l \to r \in R$. As a non-standard notion, we define a *rewrite system without overlaps* to be a ground rewrite system R that is ordered by $\succ$, and whenever $l \to r \in R$ then there is no other rule in R of the form $s[l] \to t$ or $s \to t[l]$. In other words, no rule can be reduced by another rule, neither the left hand side *nor the right hand side*. Any rewrite system without overlaps is a convergent ground rewrite system. In the sequel, the letter R will always denote a (ground) rewrite system without overlaps.

We show how every non-contradictory context Λ induces a ground rewrite system R_Λ without overlaps. The general technique is taken from the completeness proof of the superposition calculus [BG98, NR01] but adapted to our needs.

First, for a given non-contradictory context Λ and positive ground Σ-equation $s \approx t$ we define by induction on the literal ordering $\succ$ sets of rewrite rules $\epsilon^\Lambda_{s\approx t}$ and $R^\Lambda_{s\approx t}$ as follows. Assume that $\epsilon^\Lambda_{s'\approx t'}$ has already been defined for all ground Σ-equations $s' \approx t'$ with $s \approx t \succ s' \approx t'$. Where $R^\Lambda_{s\approx t} = \bigcup_{s\approx t\succ s'\approx t'} \epsilon^\Lambda_{s'\approx t'}$, define

$$\epsilon^\Lambda_{s\approx t} = \begin{cases} \{s \to t\} & \text{if } I_\Lambda \models s \approx t,\ s \succ t, \text{ and } s \text{ and } t \text{ are irreducible wrt. } R^\Lambda_{s\approx t} \\ \emptyset & \text{otherwise} \end{cases}$$

Then, $R_\Lambda = \bigcup_{s\approx t} \epsilon^\Lambda_{s\approx t}$ where s and t range over all ground Σ-terms.

By construction, R_Λ has no critical pairs, neither with left hand sides nor with right hand sides, and thus is a rewrite system without overlaps. Since $\succ$ is a well-founded ordering, R_Λ is a convergent rewrite system by construction. The given context Λ comes into play as stated in the first condition of the definition of $\epsilon^\Lambda_{s\approx t}$, which says, in other words, that Λ must produce $s \approx t$ as a necessary condition for $s \to t$ to be contained in R_Λ. An important detail is that whenever Λ is non-contradictory and produces $s \approx t$, then it will also produce $t \approx s$. Thus, if $s \prec t$ then $s \approx t$ may still be turned into the rewrite rule $t \to s$ in R_Λ by means of its symmetric version $t \approx s$.

It is well known that for any convergent ground rewrite system R, and any two terms s and t, $R \models_{\mathrm{E}} s \approx t$ if and only if there is a term u such that $s \to_R^\star u$ and $t \to_R^\star u$. This result thus applies in particular to ground rewrite systems without overlaps.

Where the $\mathcal{ME}$ calculus would associate to a sequent $\Lambda \vdash \Phi$ the interpretation I_Λ as a candidate model of Φ, the $\mathcal{ME}_{\mathrm{E}}$ calculus will instead associate to it the E-interpretation R_Λ^{E}, the congruence closure of R_Λ (or, more correctly, of the interpretation containing the same equations as R_Λ). There is an interesting connection between the two interpretations: if L is a ground literal and $L{\downarrow_{R_\Lambda}}$ is the normal form of L wrt. R_Λ then $R_\Lambda^{\mathrm{E}} \models L$ (or, equivalently, $R_\Lambda \models_E L$) iff $I_\Lambda \models L{\downarrow_{R_\Lambda}}$ or $L{\downarrow_{R_\Lambda}}$ is a trivial equation. This connection is fundamental to $\mathcal{ME}_{\mathrm{E}}$, as it makes it possible to reduce satisfiability in the intended E-interpretation R_Λ^{E} to satisfiability in I_Λ.

Observe that if Λ is say, a set of parameter-free literals, then R_Λ, even if convergent, maybe incomplete wrt the equational theory presented by the positive equations of Λ. For instance, with $\Lambda = \{\neg v, a \approx b, b \approx c\}$ and the ordering $a \succ b \succ c$ the induced rewrite system $R_\Lambda = \{b \to c\}$ is clearly incomplete wrt. the equational theory $\{a \approx b, b \approx c\}$. In general then it might be necessary to add enough equations to a context Λ to make R_Λ complete. As we will see, the $\mathcal{ME}_{\mathrm{E}}$ calculus does need that, and it achieves it not by saturating a context directly, but by saturating its corresponding clause set.

For another example for the model construction let $\Lambda = \{a \approx u, b \approx c, a \not\approx c\}$ a non-contradictory context. With the ordering $a \succ b \succ c$ the induced rewrite system R_Λ is again $\{b \to c\}$. To see why, observe that the candidate rule $a \to c$ is assigned false by I_Λ, as Λ does not produce $a \approx c$, and that the other candidate $a \to b$ is reducible by the smaller rule $b \to c$. Had we chosen to omit in the definition of ϵ the condition "t is irreducible wrt $R^\Lambda_{s \approx t}$" [8] the construction would have given $R_\Lambda = \{a \to b, b \to c\}$. This leads to the undesirable situation that a constrained clause, say, $a \not\approx c \cdot \emptyset$ is falsified by R_Λ^{E}. But the $\mathcal{ME}_{\mathrm{E}}$ calculus cannot modify Λ to revert this situation, and to detect the inconsistency (ordered) paramodulation into variables would be needed.

4.4 Semantics of Constrained Clauses, Redundancy and Saturation

Let $C \cdot \Gamma$ be a ground constrained clause and R a ground rewrite system. We say that R is an *E-model* of $C \cdot \Gamma$ and write $R \models_{\mathrm{E}} C \cdot \Gamma$ iff $\Gamma \not\subseteq R$ or $R \models_{\mathrm{E}} C$ (in the sense of Section 3, by treating R as an interpretation). We write $R \models_{\mathrm{E}} \Phi$ for a set Φ of constrained clauses iff $R \models_{\mathrm{E}} C \cdot \Gamma$ for all $C \cdot \Gamma \in \Phi$. If F is a non-ground constrained clause (set) we write $R \models_{\mathrm{E}} F$ iff $I \models_{\mathrm{E}} F^{\mathrm{gr}}$.

The overloading of the terminology and the notation here should cause no confusion in the rest of the paper, because it will alway be clear from context whether we are talking about ordinary clauses or constrained ones. The general intuition for this notion of satisfiability for constrained clauses is that ground constrained clauses whose constraint is not a subset of a rewrite system R are considered to be trivially satisfied by R, while the other constrained clauses are considered to be satisfied by R exactly when their

[8] This condition is absent in the model construction for the superposition calculus. Its presence in the end explains why paramodulation into smaller sides of equations is necessary.

non-constraint part is E-satisfied by R. Note that for constrained clauses $C \cdot \emptyset$ with an empty constraint, $R \models_{\mathrm{E}} C \cdot \emptyset$ iff $R \models_{\mathrm{E}} C$.

If Φ and Φ' are sets of constrained clauses, we say that Φ *entails* Φ' *wrt.* R, written as $\Phi \models_R \Phi'$, iff $R \models_{\mathrm{E}} \Phi$ implies $R \models_{\mathrm{E}} \Phi'$.

Let Φ be a set of constrained clauses and $C \cdot \Gamma$ a ground constrained clause. Define $\Phi_{C\cdot\Gamma} = \{C' \cdot \Gamma' \in \Phi^{\mathrm{gr}} \mid C' \cdot \Gamma' \prec C \cdot \Gamma\}$ as the set of ground instances of clauses from Φ that are smaller than $C \cdot \Gamma$.

Let R be a rewrite system without overlaps. We say that the ground constrained clause $C \cdot \Gamma$ is *redundant wrt.* Φ *and* R iff $\Phi_{C\cdot\Gamma} \models_R C \cdot \Gamma$, that is, iff $C \cdot \Gamma$ is entailed wrt. R by smaller ground instances of clauses from Φ. Notice that if $\Gamma \not\subseteq R$ then $C \cdot \Gamma$ is trivially redundant wrt. every constrained clause set and R (as R is ordered by $\succ$). For a (possibly non-ground) constrained clause $C \cdot \Gamma$ we say that $C \cdot \Gamma$ is *redundant wrt.* Φ *and* R iff all ground instances of $C \cdot \Gamma$ are redundant wrt. Φ and R.

Suppose $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is a ground inference, for some constrained clause $C' \cdot \Gamma'$, where D stands for $\mathsf{Ref}(\epsilon)$ or $\mathsf{Para}(l \approx r, \epsilon)$ (with $l \approx r$ ground). The ground inference is called *redundant wrt.* Φ *and* R iff $\Phi_{C\cdot\Gamma} \models_R C' \cdot \Gamma'$. We say that a Ref or Para inference is *redundant wrt.* Φ *and* R iff every ground instance of it is redundant wrt. Φ and R.

Let Λ be a context. Let $R^{\Lambda}_{s\approx t} = \bigcup_{s\approx t \succ s'\approx t'} \epsilon^{\Lambda}_{s'\approx t'}$ be the rewrite system defined earlier and consisting of those ground rules true in I_Λ that are smaller than $s \approx t$.

Definition 4.1 (Productive constrained clause)
Let $C \cdot \Gamma = A_1 \vee \cdots \vee A_m \cdot \Gamma$ be a ground constrained clause, for some $m \geq 0$, where A_i is a positive non-trivial equation for all $i = 1, \ldots, m$. We say that $C \cdot \Gamma$ is *productive wrt.* Λ iff $\Gamma \subseteq R_\Lambda$ and A_i is irreducible wrt. $R^{\Lambda}_{A_i}$ for all $i = 1, \ldots, m$. A (possibly non-ground) constrained clause $C \cdot \Gamma$ is *productive wrt.* Λ iff some ground instance of $C \cdot \Gamma$ is productive wrt. Λ. □

Notice that $R^{\Lambda}_{s\approx t} = R^{\Lambda}_{t\approx s}$ for any ground terms s and t. This follows from the fact that equations are compared by taking the multisets of their top-level equations. Whether some A_i is written $s \approx t$ or $t \approx s$ thus is irrelevant.
Intuitively, if $C \cdot \Gamma$ is a productive ground constrained clauses wrt. Λ then C provides positive equations, all irreducible in the sense as stated, at least one of which must be satisfied by I_Λ, so that in consequence R^{E}_{Λ} satisfies $C \cdot \Gamma$. The following definition turns this intuition into a demand on Λ (in its second item).

Definition 4.2 (Saturation up to redundancy)
A sequent $\Lambda \vdash \Phi$ is *saturated up to redundancy* iff for all $C \cdot \Gamma \in \Phi$ such that $C \cdot \Gamma$ is not redundant wrt. Φ and R_Λ, the following hold:

1. For every inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$, where D stands for $\mathsf{Ref}(\sigma)$ or $\mathsf{Para}(l \approx r, \sigma)$ with a parameter-free $l \approx r \in_\sim \Lambda$, the clause $(C \cdot \Gamma)\sigma$ is redundant wrt. Φ and R_Λ or the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. Φ and R_Λ.

2. For every grounding substitution γ for $C \cdot \Gamma$, if $C \neq \Box$ and $(C \cdot \Gamma)\gamma$ is productive wrt. Λ and non-redundant wrt. Φ and R_Λ, then $I_\Lambda \models C\gamma$.

□

Referring back to our informal explanation of the calculus, and ignoring the redundancy concepts in Definition 4.2, ground instances of constrained clauses that are not productive wrt. Λ are subject to the first condition. It requires a sufficient number of applications of the Ref and Para rules to reduce (lifted versions of) such constrained clauses to constrained clauses productive wrt. Λ. The equality reasoning rules in $\mathcal{ME}_{\mathrm{E}}$, which are based on Ref and Para, together with the Split rule, all defined in the next section, make sure that both conditions will be met in the limit of a derivation.

The next proposition clarifies under what conditions R^{E}_{Λ} is a model for all constrained clauses Φ in a sequent $\Lambda \vdash \Phi$ saturated up to redundancy.[9]

Proposition 4.3
Let $\Lambda \vdash \Phi$ be a sequent saturated up to redundancy and suppose Φ is a constrained clause set without expansion constraints. Then, $R_\Lambda \models_{\mathrm{E}} \Phi$ if and only if Φ contains no constrained clause of the form $\square \cdot \Gamma$ that is productive wrt. Λ and non-redundant wrt. Φ and R_Λ.

Notice that Proposition 4.3 applies to a *statically* given sequent $\Lambda \vdash \Phi$. The connection to the *dynamic* derivation process of the $\mathcal{ME}_{\mathrm{E}}$ calculus will be given later, and Proposition 4.3 will be essential then in proving the correctness of the $\mathcal{ME}_{\mathrm{E}}$ calculus.

5 $\mathcal{ME}_{\mathrm{E}}$ Calculus

Like its predecessor, the $\mathcal{ME}_{\mathrm{E}}$ calculus consists of a few basic derivation rules and a number of optional ones meant to improve the performance of implementations of the calculus. The basic derivation rules include rules for equality reasoning and two rules, namely Split and Close, which are not specific to the theory of equality. We start with a description of the basic rules.

5.1 Derivation Rules for Equality Reasoning

The following two rules $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$ are the only mandatory rules specific to the theory of equality. They extend the Ref and Para derivation rules of Section 4 in a straightforward way to sequents.

$$\mathsf{Ref}_{\mathcal{ME}}(\sigma)\quad \frac{\Lambda \vdash \Phi,\ C \cdot \Gamma}{\Lambda \vdash \Phi,\ C \cdot \Gamma,\ C' \cdot \Gamma'} \quad \text{if} \begin{cases} C \cdot \Gamma \Rightarrow_{\mathsf{Ref}(\sigma)} C' \cdot \Gamma', \text{ and} \\ \Phi \cup \{C \cdot \Gamma\} \text{ contains no variant of } C' \cdot \Gamma'. \end{cases}$$

$$\mathsf{Para}_{\mathcal{ME}}(l \approx r, \sigma)\quad \frac{\Lambda \vdash \Phi,\ C \cdot \Gamma}{\Lambda \vdash \Phi,\ C \cdot \Gamma,\ C' \cdot \Gamma'} \quad \text{if} \begin{cases} l \approx r \text{ is a parameter-free fresh variant} \\ \text{of a } \Sigma\text{-equation in } \Lambda, \\ C \cdot \Gamma \Rightarrow_{\mathsf{Para}(l \approx r, \sigma)} C' \cdot \Gamma', \text{ and} \\ \text{no variant of } C' \cdot \Gamma' \text{ is in } \Phi \cup \{C \cdot \Gamma\}. \end{cases}$$

[9] Proofs can be found in the appendix.

Observe that all initial constrained clauses are built over Σ and that Simp below can add only constrained Σ-clauses. Since $l \approx r$ in $\mathsf{Para}_{\mathcal{ME}}(l \approx r, \sigma)$ is a Σ-equation, it follows that all constrained clauses derivable in the calculus are built over Σ.

The purpose of both the $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$ rules is to reduce the question of satisfiability of a constrained clause in the intended E-interpretation $R^{\mathrm{E}}_{\Lambda_{\mathbf{B}}}$, where $\Lambda_{\mathbf{B}}$ is a certain limit context (cf. Section 6), to deriving a smaller one and answering the question wrt. that one. Notice that constraints have a rather passive rôle in both derivation rules. In particular, Para is not applicable to constraints. The requirement in $\mathsf{Para}_{\mathcal{ME}}$ that $l \approx r$ be a *parameter-free variant* of an equation in the context guarantees that all constrained clause sets derivable by the calculus are parameter-free.

In both rules, the test for $C' \cdot \Gamma'$ being not contained in $\Phi \cup \{C \cdot \Gamma\}$ is needed in interplay with deletion of constrained clauses based on non-proper subsumption (see Simp below). Without this test, it is conceivable the calculus derives a sequence of constrained clause sets[10] $\{P(x) \approx \mathbf{t}, \ldots\}, \{P(x) \approx \mathbf{t}, P(y) \approx \mathbf{t}, \ldots\}, \{P(y) \approx \mathbf{t}, \ldots\}, \{P(x) \approx \mathbf{t}, P(y) \approx \mathbf{t}, \ldots\}, \{P(x) \approx \mathbf{t}, \ldots\}, \ldots$. Notice that neither the clause $P(x) \approx \mathbf{t}$ nor $P(y) \approx \mathbf{t}$ is persistent. The problem with such situations is there is no "well-founded" way to argue in the completeness proof that $P(x) \approx \mathbf{t}$ will be satisfied by the candidate model.

5.2 Basic Derivation Rules

The mandatory rules Split and Close below are taken from the $\mathcal{ME}$ calculus without equality. In fact, these rules as presented in [BT03a] could be taken *without any change* for integration in $\mathcal{ME}_{\mathrm{E}}$. It just requires to translate the constrained clauses handled by $\mathcal{ME}_{\mathrm{E}}$ to clausal form before applying the rules. However, we found the presentation of $\mathcal{ME}_{\mathrm{E}}$ more clear by instead defining adapted versions of these rules. The translation to clausal form thus will be implicitly used then. Both the Split and Close rule are based on the concept of a *context unifier*.

Definition 5.1 (Context Unifier)
Let Λ be a context and $C = L_1 \vee \cdots \vee L_m$ an ordinary clause. A substitution σ is *a context unifier of C against Λ* iff there are fresh p-variants $K_1, \ldots, K_m \in_{\simeq} \Lambda$ such that σ is a most general simultaneous unifier of the sets $\{K_1, \overline{L_1}\}, \ldots, \{K_m, \overline{L_m}\}$.

For each $i = 1, \ldots, m$, we say that a literal $K'_i \in \Lambda$ is a *context literal of σ* if $K'_i \simeq K_i$, and that $L_i\sigma$ is a *remainder literal of σ* if $(\mathcal{P}ar(K_i))\sigma \not\subseteq V$. We say that σ is *productive* iff K_i produces $\overline{L_i}\sigma$ in Λ for all $i = 1, \ldots, m$. □

A context unifier σ of C against Λ is *admissible (for* Split*)* iff every remainder literal L of σ is parameter- or variable-free and for all distinct remainder literals L and K of σ $\mathcal{V}ar(L) \cap \mathcal{V}ar(K) = \emptyset$.

$$\mathsf{Split}(L, \sigma) \quad \frac{\Lambda \vdash \Phi,\ C \cdot \Gamma}{\Lambda,\ L \vdash \Phi,\ C \cdot \Gamma \qquad \Lambda,\ \overline{L}^{\mathrm{sko}} \vdash \Phi,\ C \cdot \Gamma} \quad \text{if } \star$$

[10] The constraint parts all are, say, $\emptyset$, and are not written.

where $\star$ is

1. $C = A_1 \vee \cdots \vee A_m$ with $m \geq 0$ and for all $i = 1, \ldots, m$, A_i is a positive non-trivial equation,
2. σ is an admissible context unifier of $(C \cdot \Gamma)^c$ against Λ with remainder literal L, and
3. neither L nor $\overline{L}^{\mathrm{sko}}$ is contradictory with Λ.

A Split inference is *productive* iff σ is a *productive* context unifier of $(C \cdot \Gamma)^c$ against Λ.

To obtain a complete calculus Split needs to be applied only when $C \cdot \Gamma$ has an R_Λ-irreducible ground instance that is falsified by the E-interpretation R^{E}_Λ. Technically, these ground instances are approximated by the productive ones, in terms of Definition 4.1, and a productive context unifier is guaranteed to exist then. Applying a Split inference then will modify the context so that it E-satisfies such a ground instance afterwards, which marks some progress in the derivation.

$$\mathsf{Close}(\sigma) \quad \frac{\Lambda \vdash \Phi, C \cdot \Gamma}{\Lambda \vdash \square \cdot \emptyset} \quad \text{if} \begin{cases} \Phi \neq \emptyset \text{ or } C \cdot \Gamma \neq \square \cdot \emptyset, \text{ and} \\ \sigma \text{ is a context unifier of } (C \cdot \Gamma)^c \text{ against } \Lambda \\ \text{with no remainder literals.} \end{cases}$$

The purpose of the Close rule is to detect a trivial inconsistency between the context and a constrained clause.

5.3 Optional Derivation Rules

The first optional derivation rule, Compact simplifies a context by removing a superfluous literal.

$$\mathsf{Compact} \quad \frac{\Lambda, K, L \vdash \Phi}{\Lambda, L \vdash \Phi} \quad \text{if } K \geq L$$

Like DPLL, the $\mathcal{ME}$ calculus includes an optional derivation rule, called Assert, to insert a literal into a context without causing branching. In $\mathcal{ME}$ this rule bears close resemblance to the unit-resulting resolution rule. In the terminology of the present paper, where constrained clauses instead of clauses are used, the $\mathcal{ME}$Assert rule can be rephrased as follows:

$$\mathsf{Assert}_{\mathcal{ME}}(\sigma) \quad \frac{\Lambda \vdash \Phi,\ C \vee L \cdot \Gamma}{\Lambda,\ L\sigma \vdash \Phi,\ C \vee L \cdot \Gamma} \quad \text{if} \begin{cases} \sigma \text{ is a context unifier of } C \cdot \Gamma \text{ against } \Lambda \\ \text{without remainder,} \\ L\sigma \text{ is non-contradictory with } \Lambda, \text{ and} \\ \text{there is no } K \in_\simeq \Lambda \text{ such that } K \geq L\sigma. \end{cases}$$

This rule could be used without change for $\mathcal{ME}_{\mathrm{E}}$. However, as it is an *optional* rule, there is no good reason not to strengthen it, even if the applicability conditions are no

longer decidable and can only be approximated. Therefore we propose the following rather general Assert below.

To define the Assert rule, some more preliminaries are needed.

Let us fix a constant a from the signature $\Sigma^{\mathrm{sko}} \setminus \Sigma$ and consider the substitution $\alpha := \{v \mapsto a \mid v \in V\}$. [11] Given a literal L, we denote by L^a the literal $L\alpha$. Note that L^a is ground if, and only if, L is variable-free. Similarly, given a context Λ, we denote by Λ^a the set of *unit clauses* obtained from Λ by removing the pseudo-literal $\neg v$, replacing each literal L of Λ with L^a, and considering it as a unit clause.[12]

$$\mathsf{Assert}(L) \quad \frac{\Lambda \vdash \Phi}{\Lambda,\ L \vdash \Phi} \quad \text{if} \begin{cases} \Lambda^a \cup \Phi^c \models_{\mathrm{E}} L^a, \\ L \text{ is non-contradictory with } \Lambda, \text{ and} \\ \text{there is no } K \in_{\simeq} \Lambda \text{ such that } K \geq L. \end{cases}$$

As an example, Assert is applicable to the sequent $\neg v, P(u,b) \approx \mathbf{t}, b \approx c \vdash P(x,y) \not\approx \mathbf{t} \vee f(x) \approx y \cdot \emptyset$ to yield the new context equation $f(u) \approx c$.

The third condition of Assert avoids the introduction of superfluous literals in the context. The first condition is needed for soundness. This condition is not decidable in its full generality and so can only be approximated with an incomplete test. This, however, is not a problem given that Assert is an optional rule in $\mathcal{ME}_{\mathrm{E}}$.

The following result implies that $\mathsf{Assert}_{\mathcal{ME}}$ is a special case of Assert. Hence, $\mathsf{Assert}_{\mathcal{ME}}$ can be used within $\mathcal{ME}_{\mathrm{E}}$ as a computationally cheap, yet practically effective, derivation rule.

Proposition 5.2
Let Λ be a context, Φ a set of constrained clauses and $C \vee L \cdot \Gamma \in \Phi$ a constrained clause. If σ is a context unifier of $C \cdot \Gamma$ against Λ without remainder, then $\Lambda^a \cup \Phi^c \models_{\mathrm{E}} (L\sigma)^a$.

Proof. Let σ be a context unifier of $C \cdot \Gamma$ against Λ without remainder. We directly show $\Lambda^a \cup \Phi^c \models_{\mathrm{E}} (L\sigma)^a$. Let $(C \vee L \cdot \Gamma)^c = L_1 \vee \cdots \vee L_m \vee L_{m+1} \vee \cdots \vee L_n \vee L$, for some $n \geq m \geq 0$, where, for uniformity of notation $L_j := s_j \not\approx t_j$, for all $j = m+1, \ldots, n$, where $\Gamma = \{s_{m+1} \to t_{m+1}, \ldots, s_n \to t_n\}$.

By definition of context unifier, there are fresh $K_1, \ldots, K_n \in_{\simeq} \Lambda$ such that σ is a simultaneous unifier of $\{\{K_1, \overline{L_1}\}, \ldots, \{K_n, \overline{L_n}\}\}$ and $(\mathcal{P}ar(K_i))\sigma \subseteq V$ for all $i = 1, \ldots, n$. With $(\mathcal{P}ar(K_i))\sigma \subseteq V$ and L_i is parameter-free for each i, it is easy to see that σ^a is a simultaneous unifier of $\{\{K_1^a, \overline{L_1}\}, \ldots, \{K_n^a, \overline{L_n}\}\}$. Since $K_i^a \in_{\simeq} \Lambda^a$ for each i, it follows from the soundness of (hyper-)resolution $\Lambda^a \cup \{(C \vee L \cdot \Gamma)^c\} \models_{\mathrm{E}} L\sigma^a$. From $C \vee L \cdot \Gamma \in \Phi$, as given, it follows $\Lambda^a \cup \Phi^c \models_{\mathrm{E}} L\sigma^a$. With $L\sigma^a = (L\sigma)^a$ the result follows immediately. □

[11] Strictly speaking, α is not a substitution in the standard sense because $\mathcal{D}om(\alpha)$ is not finite. But this will cause no problems here.

[12] Here and below Φ^c denotes the set of clausal forms of all constrained clauses in Φ.

5.4 Simplification

The Simp derivation rule defined below is an optional derivation rule. Its purpose is to replace a constrained clause by a *simpler* one. The Simp rule is general enough to accomodate the simplification rules of $\mathcal{ME}$ (Except for the Subsume rule) and also various new simplification rules connected with equality. There are too many specific characterizations of when constrained clauses are redundant or can be simplified. Instead of attempting to define individual derivation rules covering specific situations we provide a generic simplification rule and discuss some of its instantiations. To formulate it, we need one more prerequisite.

Definition 5.3 (Compatible rewrite system)
Let Λ be a context and R a (ground) rewrite system without overlaps. We say that R is *compatible with* Λ iff there is no rule $l \to r \in R$ and no negative parameter-free equation $s \not\approx t \in \Lambda$ such that $s \approx t \gtrsim l \approx r$. □

The rationale behind this definition is the following. Given the *current context* Λ of a derivation, one could imagine of using its associated rewrite system R_Λ to simplify the current clause set. Unfortunately, such simplification are in general unsound because R_Λ does not grow monotonically as the context evolves. For example, suppose R_Λ contains the rule $f(a) \to a$ at some point in the derivation. If later in the derivation the equation $a \approx b$ is added to the context, with $a \succ b$, R_Λ will then contain the rule $a \to b$ but not $f(a) \to a$ anymore, as the latter rule is reducible by the former. In such a case, any simplification step based on applying $f(a) \to a$ during the derivation would be unjustified.

Consequently, in general, at any time in the derivation it is not foreseeable what the limit rewrite system will look like. Fortunately, there is a characterization in terms of Λ of what rules will necessarily *not* be included in the limit rewrite system and that can be exploited for simplification purposes: if there is a negative *parameter-free* equation $s \not\approx t \in \Lambda$, then, as the derivation proceeds, no modification Λ' of Λ can produce any (ground) instance $(s \approx t)\gamma$ (because the context Λ' would be contradictory then, which is impossible by definition of the derivaton rules). In particular, if $\Lambda_{\mathbf{B}}$ denotes the context obtained in the limit of the derivation, then $(s \approx t)\gamma$ cannot be produced by $\Lambda_{\mathbf{B}}$ and hence will not be included in $R_{\Lambda_{\mathbf{B}}}$ as a rule. In other words, $R_{\Lambda_{\mathbf{B}}}$ is compatible with any "current" context Λ of a derivation. These considerations motivate the following simplification rule.

$$\textsf{Simp} \quad \frac{\Lambda \vdash \Phi,\, C \cdot \Gamma}{\Lambda \vdash \Phi,\, C' \cdot \Gamma'} \quad \text{if } \star$$

where $\star$ is

1. $C' \cdot \Gamma' \in \Phi$ and $C' \cdot \Gamma'$ non-properly subsumes $C \cdot \Gamma$, or
2. for every rewrite system R compatible with Λ:
   (a) $C \cdot \Gamma$ is redundant wrt. $\Phi \cup \{C' \cdot \Gamma'\}$ and R,

(b) $C' \cdot \Gamma'$ is a constrained clause over Σ without expansion constraints, and

(c) $\Lambda^a \cup (\Phi \cup \{C \cdot \Gamma\})^c \models_{\mathrm{E}} (C' \cdot \Gamma')^c$.

The last condition in the definition of the Simp rule guarantees soundness.

It is not difficult to see that the explicit test for non-proper subsumption and the general redundancy test together comprise a full subsumption test.

We now discuss some specific instances of the Simp rule.

Trivial Equations. As a simple instance of the Simp rule, any constrained clause $C \cdot \Gamma$ of the form $s \approx s \vee D \cdot \Gamma$ can be simplified to $\mathbf{t} \approx \mathbf{t} \cdot \emptyset$. This simplification step actually yields the same effect as if $C \cdot \Gamma$ were deleted. Dually, any constrained clause $C \cdot \Gamma$ of the form $s \not\approx s \vee D \cdot \Gamma$ can be simplified to $D \cdot \Gamma$.

Dis-ordered Constraints. A perhaps less obvious case is when the constraint Γ of a constrained clause $C \cdot \Gamma$ contains a rule $l \to r$ such that $l \prec r$. As we assumed a liftable ordering $\succ$, $l \prec r$ implies $l\gamma \prec r\gamma$ for any ground substitution γ. By definition, no rewrite system without overlaps can contain the rule $l\gamma \to r\gamma$, i.e., $(l \to r)\gamma \notin R$, and hence $(C \cdot \Gamma)\gamma$ and also $C \cdot \Gamma$ is redundant wrt. every constrained clause set and R.

Unsatisfiable Constraints. As a simple example that takes the context into account, consider the sequent $f(x) \not\approx x \vdash a \approx b \cdot f(a) \to a$. Now, no rewrite system compatible with $\{f(x) \not\approx x\}$ can contain $f(a) \to a$. The constrained clause can therefore again be simplified to $\mathbf{t} \approx \mathbf{t} \cdot \emptyset$. Dually, in the sequent $f(x) \approx x \vdash a \approx b \cdot f(a) \to a$ the constrained clause can be simplified to $a \approx b \cdot \emptyset$. (Notice in particular that this simplification is indeed sound.)

Unit Resolution by Resolve. Another practically relevant application of Simp corresponds to applications of the unit resolution rule. Suppose a context $\Lambda, K \vdash \Phi, C \vee L \cdot \Gamma$ where K is a positive or negative equation, and suppose there is a mgu of K and $\overline{L}$ such that $(\mathcal{P}ar(K))\sigma \subseteq V$ and $(C \cdot \Gamma)\sigma = C \cdot \Gamma$. Because of $(\mathcal{P}ar(K))\sigma \subseteq V$ it follows $(\Lambda \cup \{K\})^a \cup \{(C \vee L \cdot \Gamma)^c\} \models_{\mathrm{E}} ((C \cdot \Gamma)\sigma)^c$. Together with $(C \cdot \Gamma)\sigma = C \cdot \Gamma$, the clause $C \vee L \cdot \Gamma$ can thus be simplified to $C \cdot \Gamma$. Similarly, in a sequent $\Lambda, K \vdash \Phi, C \cdot \Gamma, A$ the constrained clause $C \cdot \Gamma, A$ can be simplified to $(C \cdot \Gamma)\sigma$ when σ is an mgu of K and A and such that $(\mathcal{P}ar(K))\sigma \subseteq V$ and $(C \cdot \Gamma)\sigma = C \cdot \Gamma$. These applications of Simp have an explicit counterpart in $\mathcal{ME}$ and in DPLL, the Resolve rule. In other words, Simp covers—and generalizes—that rule.

Simplification by Constrained Clauses. As indicated above, even when orientable, context equations cannot be used in general to simplify, say, the current set of constrained clauses. However, a constrained clause comprised of an orientable positive unit clause and an empty constraint can be used to simplify (in the sense of demodulation) the clause part of a constrained clause. For instance, if $f(x) \approx x \cdot \emptyset$ and $f(y) \approx g(y) \vee f(a) \not\approx a \cdot f(y) \to a$ are among the current constrained clauses, then the latter can be simplified by two-fold demodulation with the former to obtain $y \approx g(y) \vee a \not\approx a \cdot f(y) \to a$ (which can be simplified further). Notice that such simplification steps are not restricted to input constrained clauses, as suitable unit constrained

clauses might be also obtained during a derivation by Simp inferences (for instance, by means of its instance Resolve).

Demodulation and also extended simplification techniques like contextual rewriting have been known for a long time, for instance in conjunction with the superposition calculus. Given that our model construction and the "induced" redundancy criteria are adapted from the superposition calculus, it is not too surprising that comparable simplification techniques can be obtained with our Simp rule.

5.5 Derivation Examples

The following excerpt from an $\mathcal{ME}_{\mathrm{E}}$ derivation demonstrates Para, Simp and Split in combination. It follows the example in Section 2 by taking the same context $\Lambda = \{f(u) \approx u, f(a) \not\approx a\}$. However, to be more instructive, it uses a lifted version $f(x) \approx x \vee f(f(x)) \approx b \vee f(b) \not\approx b$ of the ground clause there.

$$\neg v, \underline{f(u) \approx u}, f(a) \not\approx a \vdash \ldots, f(x) \approx x \vee \underline{f(f(x))} \approx b \vee f(b) \not\approx b \cdot \emptyset$$

$$\neg v, \underline{f(u) \approx u}, f(a) \not\approx a \vdash \ldots, \begin{array}{l} f(x) \approx x \vee f(f(x)) \approx b \vee f(b) \not\approx b \cdot \emptyset, \\ f(x) \approx x \vee \quad f(x) \approx b \quad \vee \underline{f(b)} \not\approx b \cdot f(f(x)) \to f(x) \end{array} \quad \text{(By Para)}$$

$$\neg v, \underline{f(u) \approx u}, f(a) \not\approx a \vdash \ldots, \begin{array}{l} f(x) \approx x \vee f(f(x)) \approx b \vee f(b) \not\approx b \cdot \emptyset, \\ f(x) \approx x \vee \quad f(x) \approx b \quad \vee f(b) \not\approx b \cdot f(f(x)) \to f(x) \\ f(x) \approx x \vee \quad f(x) \approx b \quad \vee \quad \underline{b \not\approx b} \quad \cdot f(f(x)) \to f(x), f(b) \to b \end{array}$$
(By Para)

$$\neg v, f(u) \approx u, f(a) \not\approx a \vdash \ldots, \begin{array}{l} f(x) \approx x \vee f(f(x)) \approx b \vee f(b) \not\approx b \cdot \emptyset, \\ f(x) \approx x \vee \quad f(x) \approx b \quad \vee f(b) \not\approx b \cdot f(f(x)) \to f(x) \\ f(x) \approx x \vee \quad f(x) \approx b \qquad\qquad \cdot f(f(x)) \to f(x), f(b) \to b \end{array}$$
(By Simp)

Among the alternatives to proceed now we focus on possible Split inferences. Consider the last sequent with the constrained clause $f(x) \approx x \vee f(x) \approx b \cdot f(f(x)) \to f(x), f(b) \to b$ and its clausal form $f(x) \approx x \vee f(x) \approx b \vee f(f(x)) \not\approx f(x) \vee f(b) \not\approx b$. Simultaneous unification of that clause literals with fresh variants of the context literals $f(a) \not\approx a, \neg v, f(u) \approx u, f(u) \approx u$, respectively, gives the (productive and admissible) context unifier $\sigma = \{x \mapsto a, \ldots\}$. The remainder literals of σ are $f(a) \approx b$, $f(f(a)) \not\approx f(a)$ and $f(b) \not\approx b$ (notice that the clause instance literal $f(a) \approx a$ is contradictory with the context and hence is a non-remainder literal). Each of them can be selected for Split. The effect of selecting $f(a) \approx b$ or $f(b) \not\approx b$ was already described in Section 2.

We turn to another sample derivation now:

$$\vdots$$

$$\neg v, \underline{P(u,u) \approx \mathbf{t}} \vdash \cdots, \underline{P(x,y) \not\approx \mathbf{t}} \vee Q(x) \approx \mathbf{t} \vee R(y) \approx \mathbf{t} \cdot \emptyset$$

$$\neg v, P(u,u) \approx \mathbf{t} \vdash \cdots, \begin{array}{r} P(x,y) \not\approx \mathbf{t} \vee Q(x) \approx \mathbf{t} \vee R(y) \approx \mathbf{t} \cdot \emptyset, \\ \underline{\mathbf{t} \not\approx \mathbf{t}} \quad \vee Q(x) \approx \mathbf{t} \vee R(x) \approx \mathbf{t} \cdot P(x,x) \to \mathbf{t} \end{array} \qquad \text{(By Para)}$$

$$\underline{\neg v}, \underline{P(u,u) \approx \mathbf{t}} \vdash \cdots, \begin{array}{r} P(x,y) \not\approx \mathbf{t} \vee Q(x) \approx \mathbf{t} \vee R(x) \approx \mathbf{t} \cdot \emptyset, \\ \underline{Q(x) \approx \mathbf{t}} \vee R(x) \approx \mathbf{t} \cdot P(x,x) \to \mathbf{t} \end{array} \qquad \text{(By Simp)}$$

$$\begin{array}{l} \neg v, P(u,u) \approx \mathbf{t}, \\ \quad Q(u) \approx \mathbf{t} \end{array} \vdash \cdots, \begin{array}{r} P(x,y) \not\approx \mathbf{t} \vee Q(x) \approx \mathbf{t} \vee R(x) \approx \mathbf{t} \cdot \emptyset, \\ Q(x) \approx \mathbf{t} \vee R(x) \approx \mathbf{t} \cdot P(x,x) \to \mathbf{t} \end{array} \qquad \text{(By Split (left))}$$

This sequence of derivation steps demonstrates one way in which the Split rule of the $\mathcal{ME}$ calculus can be simulated in $\mathcal{ME}_{\mathrm{E}}$: first, all negative clause literals are paramodulated to obtain negative trivial equations $\mathbf{t} \not\approx \mathbf{t}$, which are then eliminated with Simp (notice that Ref would eliminate these equations $\mathbf{t} \not\approx \mathbf{t}$ as well, but at the cost of introducing a new constrained clause). Then a context unifier is built with the clausal form of the resulting constrained clause. In comparison to $\mathcal{ME}$, the positive literals are still in place, while the negative literals have been instantiated and moved to the constraint part, but the context unifier will be the same. The $\mathcal{ME}$Split rule thus can be seen as a macro derivation rule in $\mathcal{ME}_{\mathrm{E}}$.

6 Correctness of the $\mathcal{ME}_{\mathrm{E}}$ Calculus

Similarly to the $\mathcal{ME}$ calculus, derivations in $\mathcal{ME}_{\mathrm{E}}$ are formally defined in terms of derivation trees. The purpose of the calculus is to build for a given clause set a derivation tree all of whose branches are failed iff the clause set is unsatisfiable. The soundness argument for the calculus is relatively straightforward and analogous to the one for the $\mathcal{ME}$ calculus. Therefore, in this section we concentrate just on completeness. A detailed soundness proof can be found in Section A.2 in the appendix.

A *derivation tree* of a set $\{C_1, \ldots, C_n\}$ of Σ-clauses is a finite tree over sequents in which the root node is the sequent $\neg v \vdash C_1 \cdot \emptyset, \ldots, C_n \cdot \emptyset$, and each non-root node is the result of applying one of the derivation rules to the node's parent.

Let $\mathbf{T}$ be a derivation tree presented as a pair $(\mathbf{N}, \mathbf{E})$, where $\mathbf{N}$ is the set of the nodes of $\mathbf{T}$ and $\mathbf{E}$ is the set of the edges of $\mathbf{T}$. A derivation $\mathcal{D} = (\mathbf{T}_i)_{i<\kappa}$ in $\mathcal{ME}_{\mathrm{E}}$ is a possibly infinite sequence of derivation trees defined in the obvious way. Each *derivation* $\mathcal{D} = ((\mathbf{N}_i, \mathbf{E}_i))_{i<\kappa}$ determines a *limit tree* $\mathbf{T} := (\bigcup_{i<\kappa} \mathbf{N}_i, \bigcup_{i<\kappa} \mathbf{E}_i)$. 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 κ nodes, and let $\Lambda_i \vdash \Phi_i$ be the sequent labeling node N_i, for all $i < \kappa$. Define $\Lambda_{\mathbf{B}} = \bigcup_{i<\kappa} \bigcap_{i \le j < \kappa} \Lambda_j$ and $\Phi_{\mathbf{B}} = \bigcup_{i<\kappa} \bigcap_{i \le j < \kappa} \Phi_j$, the sets of *persistent context literals*

and *persistent clauses*, respectively. These two sets can be combined to obtain the *limit sequent* $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ (of $\mathbf{T}$). [13]

As usual, the completeness of $\mathcal{ME}_{\mathrm{E}}$ relies on a suitable notion of fairness.

Definition 6.1 (Exhausted Branch)
Let $\mathbf{T}$ be a limit tree, and let $\mathbf{B} = (N_i)_{i<\kappa}$ be a branch in $\mathbf{T}$ with κ nodes. For all $i < \kappa$, let $\Lambda_i \vdash \Phi_i$ be the sequent labeling node N_i. The branch $\mathbf{B}$ is *exhausted* iff for each constrained clause $C \cdot \Gamma \in \Phi_{\mathbf{B}}$ that is not redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for some $j < \kappa$, all of the following hold, for all $i < \kappa$ such that $C \cdot \Gamma \in \Phi_i$:

(i) if $\mathsf{Ref}_{\mathcal{ME}}$ is applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$ and underlying Ref inference $C \cdot \Gamma \Rightarrow_{\mathsf{Ref}(\sigma)} C' \cdot \Gamma'$, and $(C \cdot \Gamma)\sigma$ is not redundant wrt. Φ_i and $R_{\Lambda_{\mathbf{B}}}$, then there is a $j < \kappa$ such that the inference $C \cdot \Gamma \Rightarrow_{\mathsf{Ref}(\sigma)} C' \cdot \Gamma'$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$.

(ii) if $\mathsf{Para}_{\mathcal{ME}}$ is applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$ and underlying Para inference $C \cdot \Gamma \Rightarrow_{\mathsf{Para}(l \approx r, \sigma)} C' \cdot \Gamma'$, where $l \approx r \in_{\sim} \Lambda_{\mathbf{B}}$ and $\Lambda_{\mathbf{B}}$ produces $(l \approx r)\sigma$, and $(C \cdot \Gamma)\sigma$ is not redundant wrt. Φ_i and $R_{\Lambda_{\mathbf{B}}}$, then there is a $j < \kappa$ such that the inference $C \cdot \Gamma \Rightarrow_{\mathsf{Para}(l \approx r, \sigma)} C' \cdot \Gamma'$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$.

(iii) if Split is applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$ and productive context unifier σ such that every context literal K of σ is a Σ-literal[14] and $K \in_{\simeq} \Lambda_{\mathbf{B}}$, and $(C \cdot \Gamma)\sigma$ is productive wrt. $\Lambda_{\mathbf{B}}$, then there is a $j < \kappa$ such that $(C \cdot \Gamma)\sigma$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$ or there is a remainder literal L of σ and a $j \geq i$ with $j < \kappa$ such that Λ_j produces L but not $\overline{L}$.

(iv) Close is not applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$ and any context unifier σ such that $K \in_{\simeq} \Lambda_{\mathbf{B}}$ for every context literal K of σ.

(v) $\Phi_i \neq \{\Box \cdot \emptyset\}$.

□

A limit tree of a derivation is *fair* iff it is a refutation tree that is, a finite tree all of whose leafs are conclusions of the Close rule, or it has an exhausted branch. A derivation is *fair* iff its limit tree is fair.

It is not too difficult to see that actually carrying out an $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ inference renders the underlying Ref or Para inference redundant *wrt. any rewrite system ordered by* $\succ$. Concerning Split, like in the $\mathcal{ME}$ calculus carrying out a Split inference also achieves what fairness demands for. These considerations indicate that a fair proof procedure indeed exists. It should not be too difficult to modify the proof procedure (and implementation) for the Model Evolution calculus described in [BFT05] accordingly.

[13] A limit sequent that does not contain the empty constrained clause $\Box \cdot \emptyset$ acts as the result of a derivation that is not a refutation.

[14] Note the restriction to Σ-literals; it is *not* possible to restrict condition (iv) in the same way.

Definition 6.1 provides a framework for fair derivations based on redundant clauses and redundant inferences. The redundancy criteria are formulated wrt. $R_{\Lambda_\mathbf{B}}$ – an object that is not available during a derivation. The redundancy tests are therefor impossible to effectively realize in their full strength. Nethertheless there are some effective and cheap sufficient redundancy tests.

- If a Split inference with a constrained clause $C \cdot \Gamma$ and context unifier σ results in a rule $l \to r$ in $\Gamma\sigma$ that in not compatible with the order $\succ$, i.e., $l \prec r$ holds, then this inference is redundant wrt. any constrained clause set and (in particular) $R_{\Lambda_\mathbf{B}}$ (as $R_{\Lambda_\mathbf{B}}$ is ordered by $\succ$).
- If in case of an $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ inference the resulting constrained clause $C' \cdot \Gamma'$ is properly subsumed by a constrained clause from the constrained clause set of the sequent it is applied to, then this inference is redundant. The case of non-proper subsumption is handled explicitly.

  Likewise, if in case of a Split inference the constrained clause instance $(C \cdot \Gamma)\sigma$ is properly subsumed by a constrained clause from the constrained clause set of the sequent it is applied to, then this inference is redundant, too.
- Any Split, $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ inference is redundant when $C\sigma$ contains a trivial positive equation $s \approx s$ or $\Gamma\sigma$ contains a a rule $l \to r$ such that $(l \not\approx r)\sigma$ is an instance of a parameter-free context literal $s \not\approx t$.

Proposition 6.2 (Exhausted branches are saturated up to redundancy)
If $\mathbf{B}$ *is an exhausted branch of a limit tree of some fair derivation then (i)* $\Lambda_\mathbf{B} \vdash \Phi_\mathbf{B}$ *is saturated up to redundancy, (ii)* $\Phi_\mathbf{B}$ *is a constrained clause set without expansion constraints, and (iii)* $\Phi_\mathbf{B}$ *contains no constrained clause of the form* $\square \cdot \Gamma$ *that is productive wrt.* $\Lambda_\mathbf{B}$ *and that is not redundant wrt.* $\Phi_\mathbf{B}$ *and* $R_{\Lambda_\mathbf{B}}$.

Propositions 6.2 and 4.3 together entail our main result:

Theorem 6.3 (Completeness of $\mathcal{ME}_\mathrm{E}$)
Let Ψ *be a parameter-free* Σ*-clause set, and* $\mathbf{T}$ *be the limit tree of a fair derivation of* Ψ. *If* $\mathbf{T}$ *is not a refutation tree, then* Ψ *is satisfiable; more specifically, for every exhausted branch* $\mathbf{B}$ *of* $\mathbf{T}$, $R_{\Lambda_\mathbf{B}} \models_\mathrm{E} \Psi$.

As an easy corollary to the soundness and completeness of the calculus we also obtain the following.

Corollary 6.4 (Bernays-Schönfinkel Class with Equality)
The $\mathcal{ME}_\mathrm{E}$ *Calculus can be used as a decision procedure for the Bernays-Schönfinkel class, i.e., for sentences with the quantifier prefix* $\exists^*\forall^*$.

7 Conclusions

We have presented the $\mathcal{ME}_{\mathrm{E}}$ calculus, an extension of the Model Evolution calculus by paramodulation-based inference rules for equality. Our main result is its correctness, in particular the completeness in combination with redundancy criteria. As for future work, we will extend the implementation of the model evolution calculus, the Darwin system [BFT05] to the $\mathcal{ME}_{\mathrm{E}}$ calculus.

There are also some theoretical issues to be addressed. The perhaps most pressing theoretical question is if or when paramodulation into smaller sides of equations can be avoided. It is clear that the current completeness proof breaks down when such inferences are no longer subject to fairness. Other questions concern further, useful instantiations of our simplification rule.

Acknowledgements. We would like to thank the reviewers for their valuable comments.

References

[Bau00] Peter Baumgartner. FDPLL – A First-Order Davis-Putnam-Logeman-Loveland Procedure. In David McAllester, editor, *CADE-17 – The 17th International Conference on Automated Deduction*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 200–219. Springer, 2000.

[BFT05] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Implementing the Model Evolution Calculus. In Stephan Schulz, Geoff Sutcliffe, and Tanel Tammet, editors, *Special Issue of the International Journal of Artificial Intelligence Tools (IJAIT)*, International Journal of Artificial Intelligence Tools, 2005. To appear.

[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.

[Bil96] Jean-Paul Billon. The Disconnection Method. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 1071 in Lecture Notes in Artificial Intelligence, pages 110–126. Springer, 1996.

[BT03a] 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.

[BT03b] Peter Baumgartner and Cesare Tinelli. The Model Evolution Calculus. Fachberichte Informatik 1–2003, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 2003.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[GK03] Harald Ganzinger and Konstantin Korovin. New directions in instance-based theorem proving. In *LICS - Logics in Computer Science*, 2003.

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

[LS02] Reinhold Letz and Gernot Stenz. Integration of Equality Reasoning into the Disconnection Calculus. In Uwe Egly and Christian G. Fermüller, editors, *TABLEAUX*, volume 2381 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2002.

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 371–443. Elsevier and MIT Press, 2001.

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

A Proofs

This appendix contains auxiliary lemmas, their proofs, and proofs of the results stated in the main part of this paper. A reference like "Lemma [BT03b]-4.12" refers to Lemma 4.12 in [BT03b], where the technical report [BT03b] is the long version of the CADE paper [BT03a]. That papers introduce the Model Evolution Calculus (without equality). We re-use results on basic properties of contexts and properties of the Split derivation rule.

Strictly speaking, the results there do not directly apply here to $\mathcal{ME}_{\mathrm{E}}$ because, first, in the present paper we adopt the convention that any context now contains with each equation also its symetric version, and, second the presentation of the Close and Split derivation rules is slightly different. However, by inspection of the proofs in [BT03b] one may convince oneself that the results there easily carry over as needed. In particular, some results are obtained by exploiting properties of the Assert and Compact derivation rules. However, the Compact rule is the same in both calculi, and the Assert rule here is defined in such a way that the properties relevant to the completeness proof are the same as in [BT03b].

A.1 Equality Reasoning on Constrained Clauses

Proposition 4.3
Let $\Lambda \vdash \Phi$ be a sequent saturated up to redundancy and suppose Φ is a constrained clause set without expansion constraints. Then, $R_\Lambda \models_{\mathrm{E}} \Phi$ if and only if Φ contains no constrained clause of the form $\Box \cdot \Gamma$ that is productive wrt. Λ and non-redundant wrt. Φ and R_Λ.

Proof. For the only-if direction suppose $R_\Lambda \models_{\mathrm{E}} \Phi$. By way of contradiction suppose that Φ contains a clause of the form $\Box \cdot \Gamma$ that is productive wrt. Λ and non-redundant wrt. Φ and R_Λ. By definition of productivity, there is a ground substitution γ for $\Box \cdot \Gamma$ such that $\Gamma\gamma \subseteq R_\Lambda$. With $R_\Lambda \models_{\mathrm{E}} \Phi$, and hence $R_\Lambda \models_{\mathrm{E}} \Box \cdot \Gamma$, it follows $R_\Lambda \models_{\mathrm{E}} \Box$, which is impossible.

Now we turn to the if-direction. By well-founded induction we will prove $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma$, for all $C \cdot \Gamma \in \Phi$ and all ground substitutions γ for $C \cdot \Gamma$. This result will directly entail $R_\Lambda \models_{\mathrm{E}} \Phi$. Hence let $C \cdot \Gamma$ be an arbitrary constrained clause from Φ and γ an arbitrary ground substitution for $C \cdot \Gamma$.

1. $(C \cdot \Gamma)\gamma$ is redundant wrt. Φ and R_Λ.
If $(C \cdot \Gamma)\gamma$ is redundant wrt. Φ and R_Λ then, by definition, $\Phi_{(C\cdot\Gamma)\gamma} \models_{R_\Lambda} (C \cdot \Gamma)\gamma$. By induction, $R_\Lambda \models_{\mathrm{E}} \Phi_{(C\cdot\Gamma)\gamma}$, and with $\Phi_{(C\cdot\Gamma)\gamma} \models_{R_\Lambda} (C \cdot \Gamma)\gamma$ it follows $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma$.

Hence assume from now on that $(C \cdot \Gamma)\gamma$ is not redundant wrt. Φ and R_Λ. Therefore, the more general constrained clause $C \cdot \Gamma$ cannot be redundant wrt. Φ and R_Λ either. Notice this is a sufficient condition to apply items (1) and (2) of Definition 4.2 below.

2. $C \cdot \Gamma = C \cdot \Gamma', A[x]$, and $x\gamma$ is reducible wrt. R_Λ.

Suppose Γ is of the form Γ', A, that A contains a variable x, i.e., $A = A[x]$, and that $x\gamma$ is reducible wrt. R_Λ. That is, $x\gamma = x\gamma[l]$ and there is a rule $l \to r \in R_\Lambda$ (for which it holds $l \succ r$).

The constraint A is a rule, say, $s \to t$, for some terms s and t. If $s\gamma \prec t\gamma$ then $(s \to t)\gamma \notin R_\Lambda$, because all rules in R_Λ are ordered from left to right. Hence $\Gamma\gamma \not\subseteq R_\Lambda$ and so $(C \cdot \Gamma)\gamma$ is redundant wrt. Φ and R_Λ and case (1) applies. Hence suppose from now on $s\gamma \succ t\gamma$.

Because Φ is given as a constrained clause set without expansion constraints, the variable x can occur only as a proper subterm of s or as a (possibly non-proper) subterm of t, or both. Therefore, l is a proper subterm of $s\gamma$ or a (possibly non-proper) subterm of $t\gamma$, or both. In any case, as a property of orderings, it follows $(s \to t)\gamma \succ l \to r$. As $(s \to t)\gamma$ contains the subterm l, it is reducible by $l \to r$. Therefore $\epsilon_{(s \approx t)\gamma} = \emptyset$, and so $(s \to t)\gamma \notin R_\Lambda$. As above, it follows $(C \cdot \Gamma)\gamma$ is redundant wrt. Φ and R_Λ and case (1) applies again.

3. $C \cdot \Gamma = L[x] \vee D \cdot \Gamma$, x does not occur in Γ and $x\gamma$ is reducible wrt. R_Λ.
Suppose C is of the form $L \vee D$ and that L contains a variable x, i.e., $L = L[x]$, and $x\gamma$ is reducible wrt. R_Λ. That is, $x\gamma = x\gamma[l]$ and there is a rule $l \to r \in R_\Lambda$. By the previous case (2) we may assume that x does not occur in any atom in Γ.

Let γ' be the substitution that is the same as γ, except for x, where we set $x\gamma' = x\gamma[r]$. That is, γ' is like γ but with the rewrite rule $l \to r$ applied to $x\gamma$. From $l \succ r$ it follows $C\gamma' \prec C\gamma$ and hence $(C \cdot \Gamma)\gamma' \prec (C \cdot \Gamma)\gamma$. By the induction hypothesis $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma'$.

We distinguish two complementary cases. If $\Gamma\gamma' \subseteq R_\Lambda$ then from $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma'$ it follows $R_\Lambda \models_{\mathrm{E}} C\gamma'$. Because of $l \to r \in R_\Lambda$ and by definition of γ' conclude with congruence $R_\Lambda \models_{\mathrm{E}} C\gamma$, which implies $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma$. If $\Gamma\gamma' \not\subseteq R_\Lambda$ then recall we assumed that x does not occur in any rule in Γ. Hence $\Gamma\gamma' = \Gamma\gamma$ and it follows $\Gamma\gamma \not\subseteq R_\Lambda$. Again, the constrained clause $(C \cdot \Gamma)\gamma$ is redundant wrt. Φ and R_Λ and case (1) applies.

4. $C \cdot \Gamma = s \not\approx t \vee D \cdot \Gamma$ and $s\gamma = t\gamma$.
If C is of the form $s \not\approx t \vee D$ and it holds $s\gamma = t\gamma$, then there is an inference $s \not\approx t \vee D \cdot \Gamma \Rightarrow_{\mathsf{Ref}(\sigma)} (D \cdot \Gamma)\sigma$, where σ is a mgu of s and t (and there is a substitution δ such that $\gamma = \sigma\delta$).

Because $(C \cdot \Gamma)\gamma$ is not redundant wrt. Φ and R_Λ, as assumed above, the more general clause $(C \cdot \Gamma)\sigma$ cannot be redundant wrt. Φ and R_Λ either. Hence, by Definition 4.2-(1), the inference $s \not\approx t \vee D \cdot \Gamma \Rightarrow_{\mathsf{Ref}(\sigma)} (D \cdot \Gamma)\sigma$ is redundant wrt. Φ and R_Λ. In particular, thus, its ground instance $(s \not\approx t \vee D \cdot \Gamma)\gamma \Rightarrow_{\mathsf{Ref}(\epsilon)} (D \cdot \Gamma)\gamma$ via γ is redundant wrt. Φ and R_Λ. By definition of redundancy, $\Phi_{(s \not\approx t \vee D \cdot \Gamma)\gamma} \models_{R_\Lambda} (D \cdot \Gamma)\gamma$. By induction, $R_\Lambda \models_{\mathrm{E}} \Phi_{(s \not\approx t \vee D \cdot \Gamma)\gamma}$, and it follows $R_\Lambda \models_{\mathrm{E}} (D \cdot \Gamma)\gamma$. Finally, from $R_\Lambda \models_{\mathrm{E}} (D \cdot \Gamma)\gamma$ it follows trivially $R_\Lambda \models_{\mathrm{E}} (s \not\approx t \vee D \cdot \Gamma)\gamma$.

5. $C \cdot \Gamma = L \vee D \cdot \Gamma$ and $L\gamma$ is reducible wrt. R_Λ at a non-variable position.
Suppose C is of the form $L \vee D$ and $L\gamma$ is reducible wrt. R_Λ. That is $L\gamma = L\gamma[l]_p$ and there is a rule $l \to r \in R_\Lambda$. If p is a position at or below a variable in L then case (2) or case (3) applies. Hence assume that p is a non-variable position in L.

We consider the ground Para inference

$$L\gamma[l]_p \vee D\gamma \cdot \Gamma\gamma \Rightarrow_{\mathsf{Para}(l\approx r,\epsilon)} L\gamma[r]_p \vee D\gamma \cdot \Gamma\gamma, l \to r \ . \tag{1}$$

By construction of R_Λ, the rewrite rule $l \to r$ written as an equation $l \approx r$ is an instance of some equation in the context Λ. Let $l' \approx r' \in_\sim \Lambda$ be a fresh variant of such an equation. Because it is fresh, we may assume γ has been extended so as to give $(l' \approx r')\gamma = l \approx r$.

Because p is a non-variable position in L, the Para inference

$$L[t]_p \vee D \cdot \Gamma \Rightarrow_{\mathsf{Para}(l'\approx r',\sigma)} (L[r] \vee D \cdot \Gamma, l' \to r')\sigma \tag{2}$$

exists, where σ is a mgu of l' and t, and $\gamma = \sigma\delta$ for some substitution δ. The ground Para inference (1) then is a ground instance of the Para inference (2) via γ.

Because $(C \cdot \Gamma)\gamma$ is not redundant wrt. Φ and R_Λ, as assumed above, the more general clause $(C \cdot \Gamma)\sigma$ cannot be redundant wrt. Φ and R_Λ either. Hence, by Definition 4.2-(1), the inference (2) is redundant wrt. Φ and R_Λ. In particular, thus, its ground instance (1) via γ is redundant wrt. Φ and R_Λ. By definition of redundancy, $\Phi_{L\gamma[l]_p \vee D\gamma\cdot\Gamma\gamma} \models_{R_\Lambda} L\gamma[r]_p \vee D\gamma \cdot \Gamma\gamma, l \to r$. By induction, $R_\Lambda \models_{\mathrm{E}} \Phi_{L\gamma[l]_p \vee D\gamma\cdot\Gamma\gamma}$, and it follows $R_\Lambda \models_{\mathrm{E}} L\gamma[r]_p \vee D\gamma \cdot \Gamma\gamma, l \to r$. Finally, from $R_\Lambda \models_{\mathrm{E}} L\gamma[r]_p \vee D\gamma \cdot \Gamma\gamma, l \to r$ and $l \to r \in R_\Lambda$, as concluded above, it follows by congruence $R_\Lambda \models_{\mathrm{E}} L\gamma[l]_p \vee D\gamma \cdot \Gamma\gamma$, i.e., $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma$.

6. $C\gamma$ is irreducible.
By the previous cases (2), (3) and (5) together we may now assume that no literal in $C\gamma$ is reducible wrt. R_Λ. We distinguish two cases.

6.1. $C \cdot \Gamma = s \not\approx t \vee D \cdot \Gamma$.
If C is of the form $s \not\approx t \vee D$ then by case (4) we may assume $s\gamma \neq t\gamma$. Because $C\gamma$ is irreducible wrt. R_Λ in particular the equation $s\gamma \neq t\gamma$ is irreducible wrt. R_Λ. Because $s\gamma \neq t\gamma$ the terms $s\gamma$ and $t\gamma$ are not joinable wrt. R_Λ. Thus, $R_\Lambda \not\models_{\mathrm{E}} s\gamma \approx t\gamma$, or, in other words $R_\Lambda \models_{\mathrm{E}} s\gamma \not\approx t\gamma$, which trivially entails $R_\Lambda \models_{\mathrm{E}} (C \cdot \Gamma)\gamma$.

6.2. $C \cdot \Gamma = s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m \cdot \Gamma$.
By case (6.1) we may assume that C does not contain a negative equation. Thus, C is of the form $s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m$, for some positive equations $s_i \approx t_i$, for all $i = 1, \ldots, m$, where $m \geq 0$. By the assumption of case (6), each equation $(s_i \approx t_i)\gamma$ is irreducible wrt. R_Λ.

We may assume $s_i\gamma \neq t_i\gamma$, because if $s_i\gamma = t_i\gamma$ then $(C \cdot \Gamma)\gamma$ is trivially redundant wrt. Φ and R_Λ and so case (1) applies. Without loss of generality assume $s_i\gamma \succ t_i\gamma$. Because $(s_i \approx t_i)\gamma$ is irreducible wrt. R_Λ it holds neither $(s_i \to t_i)\gamma \in R_\Lambda$, nor is $(s_i \approx t_i)\gamma$ reducible wrt. $R^\Lambda_{(s_i\approx t_i)\gamma}$ (all other rewrite rules in R_Λ are greater wrt. $\succ$ than $(s_i \approx t_i)\gamma$ and cannot reduce $(s_i \approx t_i)\gamma$).

We distinguish two complementary cases, where the first one will lead to a contradiction.

In the first case $\Gamma\gamma \subseteq R_\Lambda$ holds. Together with what we just concluded it follows $(C \cdot \Gamma)\gamma$ is productive wrt. Λ. Under the assumption of the if-direction of the proposition,

that Φ contains no constrained clause of the form $\square \cdot \Gamma$ that is productive wrt. Λ and that is not redundant wrt. Φ and R_Λ it follows $C \neq \square$. Consequently we have $m > 0$. By Definition 4.2-(2) it follows $I_\Lambda \models C\gamma$. Because $m > 0$ there is a j with $1 \leq j \leq m$ such that $I_\Lambda \models (s_j \approx t_j)\gamma$. From above we know that $(s_j \approx t_j)\gamma$ is irreducible wrt. $R^\Lambda_{(s_j \approx t_j)\gamma}$. But then $\epsilon^\Lambda_{(s_j \approx t_j)\gamma} = \{(s_j \approx t_j)\gamma\}$ and it follows $(s_j \approx t_j)\gamma \in R_\Lambda$ – a plain contradiction. Hence the case $\Gamma\gamma \subseteq R_\Lambda$ is impossible.

In the second case $\Gamma\gamma \not\subseteq R_\Lambda$ holds. But then $(C \cdot \Gamma)\gamma$ is redundant wrt. Φ and R_Λ and case (1) applies. □

A.2 Soundness of the $\mathcal{ME}_{\mathrm{E}}$ Calculus

An important ingredient in the soundness proof is to show that each of the $\mathcal{ME}_{\mathrm{E}}$ derivation rules preserves a particular notion of satisfiability that we call *E-a-satisfiability*. It lifts the notion of a-satisfiability in [BT03b] to the theory of equality.

Recall from Section 5.3 that if σ is a substitution, we denote by σ^a the composed substitution $\sigma\alpha$. We point out for later that for all literals L and substitutions σ such that $(\mathcal{P}ar(L))\sigma \subseteq V$ (which includes all parameter-preserving substitutions), $L\sigma^a = L^a\sigma^a$.

Recall from Section 4.1 the notion of the *clausal form* of a constrained clause $C \cdot \Gamma$, denoted by $(C \cdot \Gamma)^c$. Recall further that Φ^c denotes the set of clausal forms of all constrained clauses in Φ.

We say that a sequent $\Lambda \vdash \Phi$ is *E-a-(un)satisfiable* iff the clause set $\Lambda^a \cup \Phi^c$ is E-(un)satisfiable in the standard sense—that is, has (no) Herbrand E-models.

An essential part of the soundness proof is to show that E-a-satisfiability of sequents is preserved as one follows the sequents derived in a derivation tree. The following lemma is a precise statement.

Lemma A.2
For each rule of the $\mathcal{ME}_{\mathrm{E}}$ calculus, if the premise of the rule is E-a-satisfiable, then one of its conclusions is E-a-satisfiable as well.

Proof. We prove the claim for all $\mathcal{ME}_{\mathrm{E}}$ derivation rules, which are $\mathsf{Ref}_{\mathcal{ME}}$, $\mathsf{Para}_{\mathcal{ME}}$, Split, Close, Assert and Simp.

$\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$) We treat both cases in one. The premise of $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ has the form $\Lambda \vdash \Phi, C \cdot \Gamma$ while its conclusion has the form $\Lambda \vdash \Phi, C \cdot \Gamma, C' \cdot \Gamma'$ where $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ and D stands for $\mathsf{Ref}(\sigma)$ or $\mathsf{Para}(l \approx r, \sigma)$, where $l \approx r$ is a parameter-free (fresh) variant of a Σ-equation in Λ.[15] With the axioms of a congruence relation it is not difficult to see that $(C \cdot \Gamma)^c$ and $(C' \cdot \Gamma')^c$ are E-equivalent.

Suppose that $\Lambda \vdash \Phi, C \cdot \Gamma$ is E-a-satisfiable. By definition, $\Lambda^a \cup (\Phi \cup \{C \cdot \Gamma\})^c$ is E-satisfiable. It follows that $\Lambda^a \cup (\Phi \cup \{C \cdot \Gamma, C' \cdot \Gamma'\})^c$ also is E-satisfiable. In other

[15] That $l \approx r$ is a parameter-free (fresh) variant of a Σ-equation is not an essential prerequisite as such. It is important though that $C' \cdot \Gamma'$ is parameter-free, because E-satisfiability is defined only for parameter-free constrained clause sets. That $l \approx r$ is parameter-free guarantees that this property is preserved under $\mathsf{Para}_{\mathcal{ME}}$ inferences.

words, the sequent $\Lambda \vdash \Phi, C \cdot \Gamma, C' \cdot \Gamma'$ is E-a-satisfiable, which is the conclusion of the $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ inference.

Split) The premise of Split has the form $\Lambda \vdash \Phi$, while its conclusions have respectively the form $\Lambda, L \vdash \Phi$ and $\Lambda, \overline{L}^{\mathrm{sko}} \vdash \Phi$. Suppose that $\Lambda \vdash \Phi$ is E-a-satisfiable. Now let $\mathbf{x} := (x_1, \ldots, x_n)$ be an enumeration of all the variables of L and note that L and L^a have exactly the same variables. Then consider the unit clause L^a (or, more explicitly, $\forall \mathbf{x}\, L^a$) and its negation $\neg\forall \mathbf{x}\, L^a$. Clearly, one of the two sets

$$S_1 := \Lambda^a \cup \{L^a\} \cup \Phi^c \quad \text{and} \quad S_2 := \Lambda^a \cup \{\neg\forall \mathbf{x}\, L^a\} \cup \Phi^c$$

must be E-satisfiable. If S_1 is E-satisfiable, we have immediately that $\Lambda, L \vdash \Phi$ is E-a-satisfiable. If S_2 is E-satisfiable, then its Skolemized form $\Lambda^a \cup \{(\overline{L^a})^{\mathrm{sko}}\} \cup \Phi^c$ is also E-satisfiable. Since $(\overline{L^a})^{\mathrm{sko}} = (\overline{L}^{\mathrm{sko}})^a$, as one can easily see, we then have that $\Lambda, \overline{L}^{\mathrm{sko}} \vdash \Phi$ is E-a-satisfiable.

Close) The premise of Close has the form $\Lambda \vdash \Phi, C \cdot \Gamma$, while its conclusion has the form $\Lambda \vdash \Box \cdot \emptyset$, and there is a context unifier σ of $(C \cdot \Gamma)^c$ against Λ without remainder. As $\Lambda \vdash \Box \cdot \emptyset$ is E-a-unsatisfiable, we must show that $\Lambda \vdash \Phi, C \cdot \Gamma$ is E-a-unsatisfiable as well. We show that by proving that $\Lambda^a \cup \{(C \cdot \Gamma)^c\}$ is E-unsatisfiable.

Let $C \cdot \Gamma = L_1 \vee \cdots \vee L_m \cdot A_{m+1}, \ldots, A_n$ for some $n \geq m \geq 0$. For uniformity of notation let $L_j := s_j \not\approx t_j$, for all $j = m+1, \ldots, n$, where $A_j = s_j \to r_j$. Since σ is a context unifier of $(C \cdot \Gamma)^c$ against Λ without remainder, we know that there are fresh variants $K_1, \ldots, K_m, K_{m+1}, \ldots, K_n \in_{\simeq} \Lambda$ such that σ is a most general simultaneous unifier of $\{K_1, \overline{L_1}\}, \ldots, \{K_n, \overline{L_n}\}$ and $(\mathcal{P}ar(K_i))\sigma \subseteq V$ for all $i = 1, \ldots, n$. Let us fix the literals $K_1, \ldots, K_n$.

Clearly, σ^a is a simultaneous unifier of $\{K_1, \overline{L_1}\}$, $\ldots$, $\{K_n, \overline{L_n}\}$. By an earlier observation we know that $K_i\sigma^a = K_i^a\sigma^a$ for all $i = 1, \ldots, n$. It follows that σ^a is a simultaneous unifier of

$$\{K_1^a, \overline{L_1}\},\ \{K_2^a, \overline{L_2}\},\ \ldots,\ \{K_n^a, \overline{L_n}\}.$$

This entails that $\{K_1^a, \ldots, K_n^a, L_1 \vee \cdots \vee L_n\}$ is unsatisfiable, and hence E-unsatisfiable. From the fact that $K_1^a, \ldots, K_n^a \in_{\simeq} \Lambda^a$ it then immediately follows that $\Lambda^a \cup \{(C \cdot \Gamma)^c\}$ is E-a-unsatisfiable.

Assert) The premise of Assert has the form $\Lambda \vdash \Phi$ while its conclusion has the form $\Lambda, L \vdash \Phi$ where $\Lambda^a \cup \Phi^c \models_E L^a$.

Suppose that $\Lambda \vdash \Phi$ is E-a-satisfiable. By definition, $\Lambda^a \cup \Phi^c$ is E-satisfiable. With $\Lambda^a \cup \Phi^c \models_E L^a$ it follows that $\Lambda^a \cup \{L^a\} \cup \Phi^c$ also is E-satisfiable. In other words, the sequent $\Lambda,\ L \vdash \Phi$ is E-a-satisfiable, which is the conclusion of the Assert inference.

Simp) The premise of Simp has the form $\Lambda \vdash \Phi, C \cdot \Gamma$ while its conclusion has the form $\Lambda \vdash \Phi,\ C' \cdot \Gamma'$. In case (i), when $C \cdot \Gamma$ is removed from Φ, $C \cdot \Gamma$, the claim follows trivially. Hence suppose case (ii), where $\Lambda^a \cup (\Phi \cup \{C \cdot \Gamma\})^c \models_E (C' \cdot \Gamma')^c$.

Suppose that $\Lambda \vdash \Phi$, $C \cdot \Gamma$ is E-a-satisfiable. By definition, $\Lambda^a \cup (\Phi \cup \{C \cdot \Gamma\})^c$ is E-satisfiable. With $\Lambda^a \cup (\Phi \cup \{C \cdot \Gamma\})^c \models_E (C' \cdot \Gamma')^c$ it follows that $\Lambda^a \cup (\Phi \cup \{C' \cdot \Gamma'\})^c$ also is E-satisfiable. In other words, the sequent $\Lambda \vdash \Phi$, $C' \cdot \Gamma'$ is E-a-satisfiable, which is the conclusion of the Simp inference. □

Proposition A.3 (Soundness)
For all sets Ψ of parameter-free Σ-clauses, if Ψ has a refutation tree, then Ψ is E-unsatisfiable.

Proof. Let $\mathbf{T}_\Psi$ be a refutation tree of Ψ, where $\Psi = \{C_1, \ldots, C_n\}$ for some Σ-clauses $C_1, \ldots, C_n$, where $n \geq 0$. We prove below by structural induction that the root of any subtree of a refutation tree is E-a-unsatisfiable. This will entail in particular that $\neg v \vdash \Phi$, the root of $\mathbf{T}_\Psi$, is E-a-unsatisfiable, where $\Phi = \{C_1 \cdot \emptyset, \ldots, C_n \cdot \emptyset\}$.. The claim will then follows from the immediate facts that the sequent $\neg v \vdash \Phi$ is E-a-unsatisfiable iff Φ is E-unsatisfiable iff Ψ is E-unsatisfiable. (Notice that the latter equivalence is trivial because the constraints of all constrained clauses in Φ are empty.)

Let $\mathbf{T}$ be a subtree of a refutation tree and let N be its root. If $\mathbf{T}$ is a one-node tree, N can only have the form $\Lambda \vdash \square \cdot \emptyset$, which is trivially E-$a$-unsatisfiable. If $\mathbf{T}$ has more than one node, we can assume by induction that all the children nodes of N are E-a-unsatisfiable. But then we can conclude that N is E-a-unsatisfiable as well by the contrapositive of Lemma A.2. □

A.3 Completeness of the $\mathcal{ME}_{\mathrm{E}}$ Calculus

Lemma A.4
Let $C_1 \cdot \Gamma_1$ and $C_2 \cdot \Gamma_2$ be ground constrained clauses. If $(\Phi_j)_{C_1 \cdot \Gamma_1} \models_{R_{\Lambda_\mathbf{B}}} C_2 \cdot \Gamma_2$ for some $j < \kappa$ then $(\Phi_\mathbf{B})_{C_1 \cdot \Gamma_1} \models_{R_{\Lambda_\mathbf{B}}} C_2 \cdot \Gamma_2$.

Proof. The proof is by well-founded induction. Suppose the result to hold for all ground constrained clauses $C_1' \cdot \Gamma_1'$ and $C_2' \cdot \Gamma_2'$ such that $C_1' \cdot \Gamma_1' \prec C_1 \cdot \Gamma_1$.[16]

Suppose $(\Phi_j)_{C_1 \cdot \Gamma_1} \models_{R_{\Lambda_\mathbf{B}}} C_2 \cdot \Gamma_2$ holds for some $j < \kappa$. If $(\Phi_j)_{C_1 \cdot \Gamma_1} \subseteq (\Phi_\mathbf{B})_{C_1 \cdot \Gamma_1}$ then the result follows from the monotonicity of first-order logic with equality. Otherwise let $(\Phi_j)_{C_1 \cdot \Gamma_1}$ itself denote a finite subset of $(\Phi_j)_{C_1 \cdot \Gamma_1}$ such that the entailment in the premise holds. Such a finite set exists by compactness of first-order logic with equality.

Let $\Phi' := (\Phi_j)_{C_1 \cdot \Gamma_1} \setminus (\Phi_\mathbf{B})_{C_1 \cdot \Gamma_1}$ be those ground constrained clauses from $(\Phi_j)_{C_1 \cdot \Gamma_1}$ that are not an instance of any persisting clause in $\Phi_\mathbf{B}$. Chose any constrained clause $C' \cdot \Gamma' \in \Phi'$ arbitrary. By construction, it is a ground instance of some clause $C \cdot \Gamma \in \Phi_j$ such that $C \cdot \Gamma \notin \Phi_\mathbf{B}$. This means that $C \cdot \Gamma$ has been removed from the clause set Φ_k of a sequent $\Lambda_k \vdash \Phi_k$ labeling the node N_k of the branch $\mathbf{B}$, for some $k < \kappa$. In other words, the Simp derivation rule has been applied to $\Lambda_k \vdash \Phi_k$ with selected constrained clause $C \cdot \Gamma$. We distinguish two subcases.

[16] Thus, formally, this is induction on the lexicographic extension of the ordering $\prec$ on pairs, which compares $(C_1' \cdot \Gamma_1', C_2' \cdot \Gamma_2') \prec (C_1 \cdot \Gamma_1, C_2 \cdot \Gamma_2)$.

In the first subcase $C \cdot \Gamma$ has been removed by non-proper subsumption from Φ_k. Let $D \cdot \Delta \in \Phi_k$ be the constrained clause non-properly subsuming $C \cdot \Gamma$. As an easy inductive consequence of the definition of the $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$ derivation rules, no constrained clause set derived can contain a constrained clause and a variant of it. Hence, $D \cdot \Delta$ cannot be a variant of $C \cdot \Gamma$ and $C \cdot \Gamma$ must be a proper instance of $D \cdot \Delta$. Because the ordering based on the converse relation, proper generalization, is well-founded, by induction there is a clause $D' \cdot \Delta'$ in $\Phi_{\mathbf{B}}$ that non-properly subsumes $C \cdot \Gamma$ (it could be $D \cdot \Delta$). Now, with $C' \cdot \Gamma'$ being an instance of $C \cdot \Gamma$, $C' \cdot \Gamma'$ is an instance of $D' \cdot \Delta'$ as well. With $D' \cdot \Delta' \in \Phi_{\mathbf{B}}$, $C' \cdot \Gamma'$ thus is an instance of a persisting clause in $\Phi_{\mathbf{B}}$. With this contradiction to the construction of Φ' conclude this subcase is impossible.

Hence, as the second subcase, by definition of Simp the constrained clause $C \cdot \Gamma$, and hence its instance $C' \cdot \Gamma'$ is redundant wrt. Φ_{k+1} and $R_{\Lambda_{\mathbf{B}}}$. This conclusion makes use of the fact that $R_{\Lambda_{\mathbf{B}}}$ is compatible with Λ_k. That $R_{\Lambda_{\mathbf{B}}}$ is indeed compatible with Λ_k has been argued for after Definition 5.3 (an explicit proof on more technical grounds is based on easy results from the technical report).

That $C' \cdot \Gamma'$ is redundant wrt. Φ_{k+1} and $R_{\Lambda_{\mathbf{B}}}$ means by definition of redundancy $(\Phi_{k+1})_{C' \cdot \Gamma'} \models_{R_{\Lambda_{\mathbf{B}}}} C' \cdot \Gamma'$. With $C' \cdot \Gamma' \in \Phi' \subseteq (\Phi_j)_{C_1 \cdot \Gamma_1}$ it follows $C' \cdot \Gamma' \prec C_1 \cdot \Gamma_1$. By the induction hypothesis then

$$(\Phi_{\mathbf{B}})_{C' \cdot \Gamma'} \models_{R_{\Lambda_{\mathbf{B}}}} C' \cdot \Gamma' \ . \tag{3}$$

With $C' \cdot \Gamma' \prec C_1 \cdot \Gamma_1$ it follows easily $(\Phi_{\mathbf{B}})_{C' \cdot \Gamma'} \subseteq (\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1}$. Together with (3) and by monotonicity of first-order logic with equality it follows

$$(\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1} \models_{R_{\Lambda_{\mathbf{B}}}} C' \cdot \Gamma' \ . \tag{4}$$

Because of this entailment, the constrained clause $C' \cdot \Gamma'$ can be removed from the premise $(\Phi_j)_{C_1 \cdot \Gamma_1}$ in the given entailment at the cost of adding the stronger set $(\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1}$. More formally, from $(\Phi_j)_{C_1 \cdot \Gamma_1} \models_{R_{\Lambda_{\mathbf{B}}}} C_2 \cdot \Gamma_2$ and (4) it follows

$$(\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1} \cup ((\Phi_j)_{C_1 \cdot \Gamma_1} \setminus \{C' \cdot \Gamma'\}) \models_{R_{\Lambda_{\mathbf{B}}}} C_2 \cdot \Gamma_2 \ . \tag{5}$$

Repeating this procedure for each of the (finitely many) members of Φ' allows to conclude

$$(\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1} \cup ((\Phi_j)_{C_1 \cdot \Gamma_1} \setminus \Phi') \models_{R_{\Lambda_{\mathbf{B}}}} C_2 \cdot \Gamma_2 \ . \tag{6}$$

Recall that $\Phi' = (\Phi_j)_{C_1 \cdot \Gamma_1} \setminus (\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1}$, which implies by elementary set theory $(\Phi_j)_{C_1 \cdot \Gamma_1} \setminus \Phi' \subseteq (\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1}$. But then, $(\Phi_{\mathbf{B}})_{C_1 \cdot \Gamma_1} \models_{R_{\Lambda_{\mathbf{B}}}} C_2 \cdot \Gamma_2$ follows from (6) immediately. □

Lemma A.5
Let $C \cdot \Gamma$ be a constrained clause. If $C \cdot \Gamma$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for some $j < \kappa$ then $C \cdot \Gamma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.

Proof. Suppose $C \cdot \Gamma$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for some $j < \kappa$. Let $C' \cdot \Gamma'$ be any ground instance $C \cdot \Gamma$. By definition, $C' \cdot \Gamma'$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, which means $(\Phi_j)_{C' \cdot \Gamma'} \models_{R_{\Lambda_{\mathbf{B}}}} C' \cdot \Gamma'$. With Lemma A.4 it follows $(\Phi_{\mathbf{B}})_{C' \cdot \Gamma'} \models_{R_{\Lambda_{\mathbf{B}}}} C' \cdot \Gamma'$. In other words $C' \cdot \Gamma'$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. Because $C' \cdot \Gamma'$ was chosen as an arbitrary ground instance of $C \cdot \Gamma$, $C \cdot \Gamma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. □

Lemma A.6
Let $C \cdot \Gamma$ be a constrained clause and $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ an inference, where D stands for $\mathsf{Ref}(\sigma)$ or $\mathsf{Para}(l \approx r, \sigma)$, for some substitution σ and $l \approx r \in_{\sim} \Lambda_{\mathbf{B}}$ in the case of Para. If the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for some $j < \kappa$ then the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.

Proof. Suppose the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for some $j < \kappa$. Let γ be an arbitrary ground substitution for $C \cdot \Gamma$, and also for $l \approx r$ in the case of Para, such that $\gamma = \sigma\delta$ for some substitution δ and such that $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is a ground instance of $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ via γ, where D' stands for $\mathsf{Ref}(\epsilon)$ or $\mathsf{Para}((l \approx r)\gamma, \epsilon)$. Because chosen arbitrarily, it suffices to show that this ground instance $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.

Because the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, its instance $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$ as well. By definition of redundancy this means

$$(\Phi_j)_{(C \cdot \Gamma)\gamma} \models_{R_{\Lambda_{\mathbf{B}}}} (C' \cdot \Gamma')\delta \ . \tag{7}$$

By Lemma A.4 then

$$(\Phi_{\mathbf{B}})_{(C \cdot \Gamma)\gamma} \models_{R_{\Lambda_{\mathbf{B}}}} (C' \cdot \Gamma')\delta \ , \tag{8}$$

which, by definition, means that the ground instance $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. □

Lemma A.7 (Close Applicability)
Let $C \cdot \Gamma \in \Phi_{\mathbf{B}}$ and $i < \kappa$. If Close is applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$, then for some $j \geq i$ with $j \leq \kappa$, Close is applicable to $\Lambda_j \vdash \Phi_j$ with selected constrained clause $C \cdot \Gamma$ and such that $K \in \Lambda_{\mathbf{B}}$ for each context literal K of its context unifier.

Proof. Assume that Close is applicable to $\Lambda_i \vdash \Phi_i$ with selected clause $C \cdot \Gamma$. We will directly prove the conclusion.

Let $(C \cdot \Gamma)^c = L_1 \vee \cdots \vee L_n$ be the clausal form of $C \cdot \Gamma$, for some $n \geq 0$ and literals $L_1, \ldots, L_n$. Let σ' be the context unifier of the considered Close rule application and let $K'_1, \ldots, K'_n \in \Lambda_i$ be the context literals of σ'. With Lemma [BT03b]-4.17 it follows there are literals $K_1, \ldots, K_n \in \Lambda_{\mathbf{B}}$ such that $K_k \geq K'_k$, for $k = 1, \ldots, n$. With $C \cdot \Gamma \in \Phi_{\mathbf{B}}$ and $K_1, \ldots, K_n \in \Lambda_{\mathbf{B}}$ it follows easily that there is an $j \geq i$ with $j < \kappa$ such that $C \cdot \Gamma \in \Phi_j$ and $K_k \in \Lambda_j$, for $k = 1, \ldots, n$.

With this, it is enough to show that there is a context unifier σ of $(C \cdot \Gamma)^c$ against Λ_j without remainder and context literals $K_1, \ldots, K_n$. In the sequel the index k allways ranges from $1, \ldots, n$.

From the existence of the context unifier σ' and the fact $K_k \geq K'_k$ it follows there is a most general simultaneous unifier σ'' of $\{K_1, \overline{L_1}\}, \ldots, \{K_n, \overline{L_n}\}$. The same arguments as in the proof of the Lifting Lemma, Lemma [BT03b]-4.12, can be applied to show this. However, there is no guarantee that $(\mathcal{P}ar(K_k))\sigma'' \subseteq V$. Using the same construction as in the proof of Lemma [BT03b]-4.23, it can be shown that there is a renaming

substitution ρ such that $(\mathcal{P}ar(K_k))\sigma''\rho \subseteq V$. Setting $\sigma := \sigma''\rho$ thus gives the desired substitution. □

Proposition 6.2 (Exhausted branches are saturated up to redundancy)
If $\mathbf{B}$ *is an exhausted branch of a limit tree of some fair derivation then (i)* $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ *is saturated up to redundancy, (ii)* $\Phi_{\mathbf{B}}$ *is a constrained clause set without expansion constraints, and (iii)* $\Phi_{\mathbf{B}}$ *contains no constrained clause of the form* $\square \cdot \Gamma$ *that is productive wrt.* $\Lambda_{\mathbf{B}}$ *and that is not redundant wrt.* $\Phi_{\mathbf{B}}$ *and* $R_{\Lambda_{\mathbf{B}}}$.

Proof. Suppose $\mathbf{B}$ is an exhausted branch of a limit tree of some fair derivation. According to Definition 4.2 it suffices to chose a constrained clause $C \cdot \Gamma \in \Phi_{\mathbf{B}}$ arbitrarily and prove the properties claimed there for $C \cdot \Gamma$.

Notice the inital constrained clause set is built over the signature Σ, and so is every derivable constrained clause set by restrictions on the derivation rules that preserve this property.

If there is a $j < \kappa$ such that $C \cdot \Gamma$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, then by Lemma A.5 the constrained clause $C \cdot \Gamma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$, and nothing remains to be shown. Hence suppose from now on that $C \cdot \Gamma$ is not redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for all $j < \kappa$.

1. $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$.
This case is concerned with $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ inferences. For economy of presentation, we treat both possibilities simultaneously. Thus suppose there is an inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$, for some constrained clause $C' \cdot \Gamma'$, where D stands for $\mathsf{Ref}(\sigma)$ or $\mathsf{Para}(l \approx r, \sigma)$, for some substitution σ and $l \approx r \in_{\sim} \Lambda_{\mathbf{B}}$ in the case of Para. It suffices to show that this inference is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$ or that $(C \cdot \Gamma)\sigma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.

If there is a $j < \kappa$ such that $(C \cdot \Gamma)\sigma$ is redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, then by Lemma A.5 $(C \cdot \Gamma)\sigma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$, and nothing remains to be shown. Hence suppose from now on that $(C \cdot \Gamma)\sigma$ is not redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for all $j < \kappa$. It suffices to show that an arbitrarily chosen ground instance of the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. Hence let γ be an arbitrary ground substitution for $C \cdot \Gamma$, and also for $l \approx r$ in the case of Para, such that the range of γ consists of Σ-terms only, $\gamma = \sigma\delta$ for some substitution δ and such that $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is a ground instance of $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ via γ, where D' stands for $\mathsf{Ref}(\epsilon)$ or $\mathsf{Para}((l \approx r)\gamma, \epsilon)$. Hence we will show that this ground inference $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.

First we consider the case of Para in more detail and rule out a specific situation. More precisely, suppose $(l \to r)\gamma \notin R_{\Lambda_{\mathbf{B}}}$. Observe that by definition of Para, the equation $(l \approx r)\gamma$ used in the paramodulation inference is included in the constraint part of the derived clause, i.e., $(l \to r)\gamma \in \Gamma'\delta$ holds. But then, $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} (C' \cdot \Gamma')\delta$ follows trivially from $(l \approx r)\gamma \notin R_{\Lambda_{\mathbf{B}}}$, and so the ground inference $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is trivially redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. Hence suppose from now on $(l \to r)\gamma \in R_{\Lambda_{\mathbf{B}}}$. It follows by construction of $R_{\Lambda_{\mathbf{B}}}$ that $\Lambda_{\mathbf{B}}$ produces $(l \approx r)\gamma$. Without loss of generality

assume that $l \approx r$ itself produces $(l \approx r)\gamma$ in $\Lambda_{\mathbf{B}}$ (it could also be a different equation, but we could use it as $l \approx r$ from the start of the the proof.) For later use note that $l \approx r$ also produces $(l \approx r)\sigma$ in $\Lambda_{\mathbf{B}}$.[17]

From $C \cdot \Gamma \in \Phi_{\mathbf{B}}$ it follows there is an $i < \kappa$ such that for all $j \geq i$ with $j < \kappa$ it holds $C \cdot \Gamma \in \Phi_j$. Likewise, from $s \approx t \in \Lambda_{\mathbf{B}}$ it follows there is an i' such that for all $j' \geq i'$ it holds $l \approx r \in_{\sim} \Lambda_{j'}$. Without loss of generality assume $i \geq i'$. It follows $l \approx r \in_{\sim} \Lambda_j$, for all $j \geq i$.

Because of $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$, $\mathsf{Ref}_{\mathcal{ME}}$ or $\mathsf{Para}_{\mathcal{ME}}$ is applicable (in particular) to $\Lambda_i \vdash \Phi_i$ with selected clause $C \cdot \Gamma$, substitution σ and derived clause $C' \cdot \Gamma'$ unless $C' \cdot \Gamma'$ is contained as a variant in Φ_i. In that case, the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. $\Lambda_i \vdash \Phi_i$ and $R_{\Lambda_{\mathbf{B}}}$ and nothing remains to show.

Recall we are currently considering the case that $(C \cdot \Gamma)\sigma$ is not redundant wrt. Φ_j and $R_{\Lambda_{\mathbf{B}}}$, for every $j < \kappa$. Furthermore, as concluded above, in the case of Para we have that $\Lambda_{\mathbf{B}}$ produces $(l \approx r)\sigma$.

But then, by Definition 6.1-(i) or Definition 6.1-(ii) there is a $k < \kappa$ such that the inference $C \cdot \Gamma \Rightarrow_{\mathsf{D}} C' \cdot \Gamma'$ is redundant wrt. Φ_k and $R_{\Lambda_{\mathbf{B}}}$. By Lemma A.6 then, this inference is also redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. Therefore, in particular its instance $(C \cdot \Gamma)\gamma \Rightarrow_{\mathsf{D}'} (C' \cdot \Gamma')\delta$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$, which remained to be shown.

2. $(C \cdot \Gamma)\gamma$ *is productive wrt.* $\Lambda_{\mathbf{B}}$.
Let γ be an arbitrary grounding substitution for $C \cdot \Gamma$ such that $(C \cdot \Gamma)\gamma$ is productive wrt. $\Lambda_{\mathbf{B}}$. We will show that $R_{\Lambda_{\mathbf{B}}} \models C\gamma$ holds or that $(C \cdot \Gamma)\gamma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. Together with the sub-proof above, the proof of item (i) of the proposition statement will be complete then.

Notice we do not distinguish whether $C = \square$ or not. Hence, if $C = \square$, then, because no interpretation satisfies $\square$, we will have shown that $(\square \cdot \Gamma)\gamma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. Because γ was chosen arbitrarily, this entails that Φ contains no constrained clause of the form $\square \cdot \Gamma$ that is productive wrt. $\Lambda_{\mathbf{B}}$ and that is not redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$, which proves item (iii) of the proposition statement.

To show that $R_{\Lambda_{\mathbf{B}}} \models C\gamma$ holds or that $(C \cdot \Gamma)\gamma$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$, suppose, by way of contradiction, $R_{\Lambda_{\mathbf{B}}} \not\models C\gamma$ and that $(C \cdot \Gamma)\gamma$ is not redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.[18]

Recall that $(C \cdot \Gamma)\gamma$ is a productive constrained clause. By the syntactical constraints imposed on productive constrained clauses, $C \cdot \Gamma$ can be written as $A_1 \vee \cdots \vee A_m \cdot A_{m+1}, \ldots, A_n$ where $n \geq m \geq 0$ and $A_i\gamma$ is a non-trivial positive equation, for all $i = 1, \ldots, m$.

Since $R_{\Lambda_{\mathbf{B}}} \not\models C\gamma$, the literals $\neg A_1\gamma, \ldots, \neg A_m\gamma$ are all E-satisfied by $R_{\Lambda_{\mathbf{B}}}$. That $(C \cdot \Gamma)\gamma$ is productive wrt. $\Lambda_{\mathbf{B}}$ implies $\{A_{m+1}\gamma, \ldots, A_n\gamma\} \subseteq R_{\Lambda_{\mathbf{B}}}$. By definition of the model construction, any rewrite rule in $R_{\Lambda_{\mathbf{B}}}$ must also be produced by $\Lambda_{\mathbf{B}}$ (as an equation).

[17] The technical report contains the following lemma, from which this claim follows easily: if K produces L in Λ and $K \gtrsim K' \gtrsim L$ then K produces K' in Λ.

[18] The proof idea is the same as in the proof of Proposition 4.29 in the technical report. This connection is made explicit by taking a constrained clause $A_1 \vee \cdots \vee A_m \cdot A_{m+1}, \ldots, A_n$ as its clausal form $A_1 \vee \cdots \vee A_m \vee \neg A_{m+1} \vee \cdots \vee \neg A_n$.

Thus, $R_{\Lambda_{\mathbf{B}}}$ produces $A_{m+1}\gamma, \ldots, A_n\gamma$ (again, as equations). By Lemma [BT03b]-4.12,[19] $\Lambda_{\mathbf{B}}$ is non-contradictory, and so by Proposition [BT03b]-4.10 it follows that $\Lambda_{\mathbf{B}}$ produces $\neg A_1\gamma, \ldots, \neg A_m\gamma, A_{m+1}\gamma, \ldots, A_n\gamma$. (Here and below, we read the rules $A_{m+1}\gamma, \ldots, A_n\gamma$ again as equations.)

By the Lifting Lemma (Lemma [BT03b]-4.21), there are fresh p-variants $\neg K_1, \ldots, \neg K_m \in_{\simeq} \Lambda_{\mathbf{B}}$ and $K_{m+1}, \ldots, K_n \in_{\simeq} \Lambda_{\mathbf{B}}$ and a substitution σ such that

1. σ is a most general simultaneous unifier of
   $\{K_1, A_1\}, \ldots, \{K_m, A_m\}, \{K_{m+1}, A_{m+1}\}, \ldots, \{K_m, A_m\}$,
2. for all $k = 1, \ldots, n$, $A_k \gtrsim A_k\sigma \gtrsim A_k\gamma$,
3. for all $k = 1, \ldots, m$, $\neg K_k$ produces $\neg A_k\sigma$ in $\Lambda_{\mathbf{B}}$, and
   for all $l = m+1, \ldots, n$, K_l produces $A_l\sigma$ in $\Lambda_{\mathbf{B}}$.

Clearly, $K_1, \ldots, K_m$ must be a Σ-literal, and hence does not contain any Skolem constant, because otherwise property (2) would be impossible (we know from above that $(C \cdot \Gamma)\gamma$ is a Σ-constrained clause).

By Definition 5.1, σ is a productive context unifier of $(C \cdot \Gamma)^c$ against $\Lambda_{\mathbf{B}}$. By Lemma [BT03b]-4.22, an admissible context unifier of $(C \cdot \Gamma)^c$ against $\Lambda_{\mathbf{B}}$ can be obtained as $\sigma' = \sigma\rho$, for some renaming ρ. It is easy to see that the properties (1)-(3) stated for σ hold for σ' as well. In particular, regarding (3) observe that a literal K produces a literal L in a context Λ iff K produces a variant of L in Λ.

Also, observe that by item (2) $(C \cdot \Gamma)\gamma$ is an instance of $(C \cdot \Gamma)\sigma$. Hence, with the instance $(C \cdot \Gamma)\gamma$ being productive wrt. $\Lambda_{\mathbf{B}}$, $(C \cdot \Gamma)\sigma$ is productive wrt. $\Lambda_{\mathbf{B}}$ as well, and so is $(C \cdot \Gamma)\sigma'$. We will use this fact later.

Let $\neg K'_1, \ldots, \neg K'_m, K'_{m+1}, \ldots, K'_n \in \Lambda_{\mathbf{B}}$ be the context literals of σ'. Since $K'_k \simeq K_k$, for all $k = 1, \ldots, m$, this implies that $\neg K'_k$ produces $\neg A_k\sigma'$ in $\Lambda_{\mathbf{B}}$, for all $k = 1, \ldots, m$, and that K'_l produces $A_l\sigma'$ in $\Lambda_{\mathbf{B}}$, for all $l = m+1, \ldots, n$. By applying Lemma [BT03b]-4.20 to $\neg K'_k$ and $\neg A_k\sigma'$, for all $k = 1, \ldots, m$ and to K'_l and $A_l\sigma'$, for all $l = m+1, \ldots, n$, and taking the maximum of the indices i mentioned in the lemma's statement, we conclude that there is an i such that for all $j \geq i$ with $j < \kappa$,

$$\begin{array}{l}\text{for all } k = 1, \ldots, m,\ \neg K'_k \in \Lambda_j \text{ and } \neg K'_k \text{ produces } \neg A_k\sigma' \text{ in } \Lambda_j, \text{ and}\\ \text{for all } l = m+1, \ldots, n,\ K'_l \in \Lambda_j \text{ and } K'_l \text{ produces } A_l\sigma' \text{ in } \Lambda_j.\end{array} \tag{9}$$

By assumption, $C \cdot \Gamma$ is a constrained clause of $\Phi_{\mathbf{B}}$. Hence, there is a i' such that $C \cdot \Gamma \in \Phi_{j'}$ for all $j' \geq i'$. Without loss of generality suppose that $i \geq i'$ (otherwise i' can be used instead of i in the sequel). It follows $C \cdot \Gamma \in \Phi_j$ for all $j \geq i$.

The next goal of the proof is to show that Split is applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$ and productive context unifer σ'. To this end, we need to know first that Close is not applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$ (for any context unifier). Suppose, by way of contradiction, Close is applicable to $\Lambda_i \vdash \Phi_i$

[19] A reference like this refer to the lemma with that key in the technical report. The one or the other straightforward adaptation will be necessary, though.

with selected constrained clause $C \cdot \Gamma$. By Lemma A.7 then, Close is also applicable to Λ_j, for some $j \geq i$ with $j \leq \kappa$ and such that $K \in \Lambda_{\mathbf{B}}$ for each context literal K of the context unifier of that inference. This however is an immediate contradiction to Definition 6.1-(iv). Hence, Close is not applicable to $\Lambda_i \vdash \Phi_i$ with selected constrained clause $C \cdot \Gamma$. Definition 6.1-(v) rules out the trivial case case $C \cdot \Gamma$ is of the form $\square \cdot \emptyset$. Together this implies that all context unifiers of $(C \cdot \Gamma)^c$ against Λ_i have a non-empty remainder.

By definition, each remainder literal L of σ' is either a literal $A_k\sigma'$, for some k with $1 \leq k \leq m$ or a literal $\neg A_l\sigma'$, for some l with $m+1 \leq l \leq n$. By (9), Λ_j produces $\overline{L}$, for each remainder literal L of σ' and all $j \geq i$. By Lemma [BT03b]-4.23 then, Split is applicable to $\Lambda_i \vdash \Phi_i$ with selected clause $(C \cdot \Gamma)^c$ – or, in the notation here, with selected constrained clause $C \cdot \Gamma$ – and productive context unifier σ'.

The next subgoal of the proof is to show that Definition 6.1-(iii) applies.

Recall we assumed that $(C \cdot \Gamma)\gamma$ is not redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. With Lemma A.5 it follows that $(C \cdot \Gamma)\gamma$ is not redundant wrt. (in particular) Φ_i and $R_{\Lambda_{\mathbf{B}}}$. By definition of redundancy, the more general constrained clause $(C \cdot \Gamma)\sigma'$ is not redundant wrt. Φ_i and $R_{\Lambda_{\mathbf{B}}}$, either. Recall also from above that $(C \cdot \Gamma)\sigma'$ is productive wrt. $\Lambda_{\mathbf{B}}$. Recall further that all context literals of σ, which are $\neg K'_1, \ldots, \neg K'_m, K'_{m+1}, \ldots, K'_n$, are contained in $\Lambda_{\mathbf{B}}$, and by (9) also are contained in Λ_i.

Alltogether then, because of Definition 6.1-(iii), there is a remainder literal L of σ' and a $j \geq i$ such that Λ_j produces L but Λ_j does not produce $\overline{L}$. However, this contradicts the conclusion above, that Λ_j produces $\overline{L}$, for each remainder literal L of σ'.

Thus, the assumption that $R_{\Lambda_{\mathbf{B}}} \not\models C\gamma$ holds and that $(C \cdot \Gamma)\gamma$ is not redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$ has been refuted, and the claim follows.

$\Phi_{\mathbf{B}}$ is a constrained clause set without expansion constraints.
For the easy proof of the conclusion (ii) of the proposition statement one just needs the following observations. Initially, Φ_1 trivially is a constrained clause set without expansion constraints, because the constraint of each of its constrained clauses is empty. A new rule can be added to a constraint Γ of a constrained clause $C \cdot \Gamma$ only by an $\mathsf{Para}_{\mathcal{ME}}$ inference. Say that $l \approx r$ is the chosen equation and that σ is the mgu used in such an inference. Since the underlying Para derivation rule is restricted not to paramodulate into variables, $l\sigma$ cannot be a variable (but l can be a variable). Hence, if Γ contains no expansion rules then $(\Gamma, l \to r)\sigma$ contains no expansion rules either. (Notice here that applying a substitution to a set of constraints Γ cannot turn a non-expansion rule to a expansion rule.) Thus, any $\mathsf{Para}_{\mathcal{ME}}$ inference preserve the property of being a constrained clause without expansion constraints. The same holds for Simp inferences by definition, and trivially for $\mathsf{Ref}_{\mathcal{ME}}$ inferences. □

Lemma A.9
If $R \models_{\mathrm{E}} \Phi$ and $C \cdot \Gamma$ is redundant wrt. Φ and R then $R \models_{\mathrm{E}} C \cdot \Gamma$.

Proof. Suppose $R \models_{\mathrm{E}} \Phi$ and $C \cdot \Gamma$ is redundant wrt. Φ and R. Let $C' \cdot \Gamma'$ be any ground instance of $C \cdot \Gamma$. It suffices to show $R \models_{\mathrm{E}} C' \cdot \Gamma'$. Since $C \cdot \Gamma$ is redundant wrt. Φ and

R, by definition, its ground instance $C' \cdot \Gamma'$ is redundant wrt. Φ and R. Equivalently, $\Phi_{C' \cdot \Gamma'} \models_R C' \cdot \Gamma'$, which entails $R \models_{\mathrm{E}} C' \cdot \Gamma'$ provided $R \models_{\mathrm{E}} \Phi_{C' \cdot \Gamma'}$ holds. The latter however follows immediately from $R \models_{\mathrm{E}} \Phi$, which means $R \models_{\mathrm{E}} \Phi^{\mathrm{gr}}$ and the trivial fact $\Phi_{C' \cdot \Gamma'} \subseteq \Phi^{\mathrm{gr}}$. □

Theorem 6.3 (Completeness of $\mathcal{ME}_{\mathrm{E}}$)
Let Ψ be a parameter-free Σ-clause set, and $\mathbf{T}$ be the limit tree of a fair derivation of Ψ. If $\mathbf{T}$ is not a refutation tree, then Ψ is satisfiable; more specifically, for every exhausted branch $\mathbf{B}$ of $\mathbf{T}$ it holds $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} \Psi$.

Proof. Suppose that $\mathbf{T}$ is not a refutation tree and let $\mathbf{B}$ be an exhausted branch of $\mathbf{T}$.

By proposition 6.2 the limit sequent $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is saturated up to redundancy, and $\Phi_{\mathbf{B}}$ contains no constrained clause of the form $\square \cdot \Gamma$ that is productive wrt. $\Lambda_{\mathbf{B}}$ and that is not redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. With Proposition 4.3 it follows $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} \Phi_{\mathbf{B}}$.

To show $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} \Psi$, let C be any clause from Ψ, and it suffices to show $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} C$. By definition of derivation, $C \cdot \emptyset \in \Phi_1$. We distinguish two cases.

In the first case $C \cdot \emptyset \in \Phi_{\mathbf{B}}$. With $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} \Phi_{\mathbf{B}}$ conclude $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} C \cdot \emptyset$, which trivially implies $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} C$.

In the second case $C \cdot \emptyset \notin \Phi_{\mathbf{B}}$. Hence $C \cdot \emptyset$ has been removed at some time $k < \kappa$ from the clause set Φ_k of the sequent $\Lambda_k \vdash \Phi_k$ by an application of the Simp derivation rule. We distinguish two subcases.

In the first subcase $C \cdot \emptyset$ has been removed by non-proper subsumption from Φ_k. Let $D \cdot \emptyset \in \Phi_k$ be the constrained clause non-properly subsuming $C \cdot \emptyset$. As an easy inductive consequence of the definition of the $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$ derivation rules, no constrained clause set Φ_i derived can contain both a constrained clause and a variant of it. Hence, $D \cdot \emptyset$ cannot be a variant of $C \cdot \Phi$ and $C \cdot \Phi$ must be a proper instance of $D \cdot \emptyset$. Because the ordering based on the converse relation, proper generalization, is well-founded, by induction on this ordering there is a clause $D' \cdot \emptyset$ in $\Phi_{\mathbf{B}}$ that non-properly subsumes $C \cdot \Phi$ (it could be $D \cdot \emptyset$). Now, to $D' \cdot \emptyset$ the first case applies and, clearly, $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} D' \cdot \emptyset$ entails $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} C \cdot \emptyset$.

In the second subcase $C \cdot \emptyset$ is redundant wrt. Φ_{k+1} and $R_{\Lambda_{\mathbf{B}}}$. This conclusion makes use of the fact that $R_{\Lambda_{\mathbf{B}}}$ is compatible with Λ_k. That $R_{\Lambda_{\mathbf{B}}}$ is indeed compatible with Λ_k has been argued for after Definition 5.3.

By Lemma A.5, $C \cdot \emptyset$ is redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$. With $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} \Phi_{\mathbf{B}}$ and Lemma A.9 it follows $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} C \cdot \emptyset$, which trivially implies $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} C$. □

Corollary 6.4 (Bernays-Schönfinkel Class with Equality)
The $\mathcal{ME}_{\mathrm{E}}$ Calculus can be used as a decision procedure for the Bernays-Schönfinkel class, i.e., for sentences with the quantifier prefix $\exists^*\forall^*$.

Proof. (Sketch) It suffices to consider clause sets where the only function symbols are constants or those corresponding to the predicate symbols in the original formula. To see why any exhausted branch will be finite then, a few things have to be observed:

1. The clause part C of a constrained clause $C \cdot \Gamma$ does not grow in length by application of a derivation rule.

2. The clause part C of a constrained clause $C \cdot \Gamma$ does not grow in depth by application of a derivation rule. This is, in particular, because paramodulation into variables is forbidden.

3. The equations used for paramodulation are bound in their term depth. When $(l \approx r)\sigma$ is an instance of an equation $l \approx r$ taken from a context and used for a Para inference then $l\sigma$ and $r\sigma$ are either a variable, a constant or a functional term all arguments of which are either constants or variables. This implies that the constraint part Γ may not grow in depth.

4. However, the constraint part Γ may grow in length by application of Para. To see that this is not a problem consider two cases. In the first case the unifier σ of a Para inference identifies two variables in the clause C where an equation $l \approx r$ paramodulates into or σ binds a variable to a constant. The resulting constrained clause $(C[r] \cdot \Gamma, l \to r)\sigma$ will be smaller then in the sense that $C\sigma$ is "more instantiated" than C. Thus, there is a finite bound on the number of Para inferences of this type. In the second case σ does neither identify two variables nor replace a variable by a constant. Recall that the clause part C neither grows in length nor depth. Hence any infinite chain of Para inferences would include a repetition on the clause part. That is, at some point there is a constrained clause $C \cdot \Gamma$ and later a constrained clause $C \cdot \Gamma'$ is derived (modulo renaming). Because σ does neither identify variables nor replace a variable by a constant it follows $\Gamma \subseteq \Gamma'$ (modulo renaming). But then, it is not difficult to see that the inference leading to $C \cdot \Gamma'$ is redundant and hence, by fairness, need not be carried out. (The argumentation applies as well if $C \cdot \Gamma$ has been deleted by an application of a Simp rule.) Thus, with a suitable redundancy test any infinite sequence of the kind considered can be pruned to a finite one without sacrificing completeness.

5. As there is a depth bound on the literals in any constraint clause $C \cdot \Gamma$ derived, there is no infinite sequence of Split inferences along a branch. This is, essentially, because a context may not contain two p-variants of the same (or complementary) literal.

□