The Model Evolution Calculus with Equality

Peter Baumgartner[1] and Cesare Tinelli[2]

[1] Max-Planck Institute for Computer Science, Saarbrücken, baumgart@mpi-sb.mpg.de
[2] Department of Computer Science, The University of Iowa, tinelli@cs.uiowa.edu

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 [4] has recently been introduced by the authors of this paper as a first-order version of the propositional DPLL procedure [7]. Compared to its predecessor, the FDPLL calculus [2], 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 [3], 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 discuss 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 [1,11, 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 [4] 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) [12]. 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 [6].[3] The other is the IM described in [9]. 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 [10] 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 [9] has been extended with equality in [8]. 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. For space constraints we cannot provide proofs of the results presented in this paper. Also, we must assume that the reader has already some familiarity with the $\mathcal{ME}$ calculus. All proofs as well as a more detailed exposition the calculus can be found in the paper's extended version [5].

2 Main Ideas

The $\mathcal{ME}$ calculus of [4], 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

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

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_Λ^E 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_Λ^E 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_Λ^E 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_Λ^E 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_Λ^E.

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 $I_{R_\Lambda}^E$ 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. 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. 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 $s\sigma = t$.[4] 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 $s \gtrsim\!\!\!\!\cdot\ t$ 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. 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). An *E-interpretation* is an interpretation that is also a congruence relation on the Σ-terms. If I is an interpretation,

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

we denote by I^{E} the smallest congruence relation 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 [4], we impose for simplicity that literals in a context can contain parameters or variables but not both, but this limitation can be overcome.

Definition 3.1 (Context [4]). *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 [4], 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 $K \in_{\sim} \Lambda$ 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 [4]). *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 $\neg\nu$ 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.[5] 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}.

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.[6]

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. 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.

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

[6] 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.

Orderings. We suppose as given a reduction ordering $\succ$ that is total on ground Σ-terms. It has to be extended to rewrite rules, equations and constrained clauses. Following usual techniques [1,11, 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. (See [5] for an alternative definition.) This way, the calculus' derivation rules $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$ for equality reasoning defined in Section 5 work in an order-decreasing way.

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.

A Ref or Para inference is *ground* if both its premise and conclusion are ground and as well as the equation $l \approx r$ in the Para case. If from a given Ref or Para inference a ground inference results by applying a substitution γ to the premise, the conclusion and the used equation $l \approx r$ in case of Para, we call the resulting ground inference a *ground instance via γ (of the inference)*.

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

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 [1,11] 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 $\varepsilon^{\Lambda}_{s \approx t}$ and $R^{\Lambda}_{s \approx t}$ as

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

follows. Assume that $\varepsilon^{\Lambda}_{s'\approx t'}$ has already been defined for all ground Σ-equations $s' \approx t'$ with $s \approx t \succ s' \approx t'$. Where $R^{\Lambda}_{s\approx t} = \bigcup_{s\approx t \succ s'\approx t'} \varepsilon^{\Lambda}_{s'\approx t'}$, define

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

Then, $R_\Lambda = \bigcup_{s\approx t} \varepsilon^{\Lambda}_{s\approx t}$ 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 $\varepsilon^{\Lambda}_{s\approx t}$, 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 $t \approx s$.

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^{E}_Λ, 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 $R^{\mathrm{E}}_\Lambda \models L$ (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^{E}_Λ to satisfiability in I_Λ.

For an 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^{E}_Λ. 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.

Semantics of Constrained Clauses. 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 $R \models_{\mathrm{E}} F^{\mathrm{gr}}$.

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 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 $R \models_{\mathrm{E}} C$.

[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.

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 $R \models_{\mathrm{E}} \Phi'$.

Redundancy. 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 $\mathsf{Ref}(\varepsilon)$ or $\mathsf{Para}(l \approx r, \varepsilon)$ (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.

Saturation. Let Λ be a context. Let $R^{\Lambda}_{s\approx t} = \bigcup_{s\approx t \succ s'\approx t'} \varepsilon^{\Lambda}_{s'\approx t'}$ 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.*

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.

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.*

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.

Derivation Rules for Equality Reasoning. The following rules $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$, the only mandatory ones for equational reasoning, extend the derivation rules of Section 4 to sequents.

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

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

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.

Basic Derivation Rules. The mandatory rules Split and Close below are taken with only minor modifications from the $\mathcal{ME}$ calculus without equality [4]. This is possible because the equality reasoning is done *only* by the $\mathsf{Ref}_{\mathcal{ME}}$ and $\mathsf{Para}_{\mathcal{ME}}$ rules above. 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 $\mathcal{V}ar(L) \cap \mathcal{V}ar(K) = \emptyset$.

$$\mathsf{Split}(L,\sigma)\ \frac{\Lambda \vdash \Phi, C \cdot \Gamma}{\Lambda, L \vdash \Phi, C \cdot \Gamma \qquad \Lambda, \overline{L}^{\mathrm{sko}} \vdash \Phi, C \cdot \Gamma}\quad \text{if} \begin{cases} C = A_1 \vee \cdots \vee A_m \text{ with } m \geq 0 \\ \text{and for all } i = 1, \ldots, m, A_i \text{ is a} \\ \text{positive non-trivial equation, } \sigma \\ \text{is an admissible context unifier} \\ \text{of } (C \cdot \Gamma)^c \text{ against } \Lambda \text{ with} \\ \text{remainder literal } L, \text{ and neither} \\ L \text{ nor } \overline{L}^{\mathrm{sko}} \text{ is contradictory with} \\ \Lambda. \end{cases}$$

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.

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

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

Optional Derivation Rules. 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. The $\mathcal{ME}_{\mathrm{E}}$ calculus has a suitable version of the Assert rule which is also more general than the one in $\mathcal{ME}$. To define it we need some more preliminaries first.

Let us fix a constant a from the signature $\Sigma^{\mathrm{sko}} \setminus \Sigma$ and consider the substitution $\alpha := \{v \mapsto a \mid v \in V\}$. 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.[9]

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

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

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. This, however, is not a problem given that Assert is an optional rule in $\mathcal{ME}_{\mathrm{E}}$. See [5] for an explanation of how the Assert rule of $\mathcal{ME}$ (with its concrete preconditions) can be seen as a special case of Assert above.

Simplification. The purpose of simplification is to replace a constrained clause by a *simpler* one. The optional Simp rule below is general enough to accomodate the simplification rules of $\mathcal{ME}$ [10] and also various new simplification rules connected with equality. To formulate it we need one more notion.

For any context Λ, a (ground) rewrite system R without overlaps is *compatible with* Λ iff there is no $l \to r \in R$ and no parameter-free $s \not\approx t \in \Lambda$ such that $s \approx t \gtrsim l \approx r$.

$$\mathsf{Simp}\ \frac{\Lambda \vdash \Phi, C \cdot \Gamma}{\Lambda \vdash \Phi, C' \cdot \Gamma'}\ \text{if}\ \begin{cases} \text{(i)} & C' \cdot \Gamma' \in \Phi \text{ and } C' \cdot \Gamma' \text{ non-properly subsumes } C \cdot \Gamma, \text{ or} \\ \text{(ii)} & \text{for every rewrite system } R \text{ compatible with } \Lambda: \\ & C \cdot \Gamma \text{ is redundant wrt. } \Phi \cup \{C' \cdot \Gamma'\} \text{ and } R, \\ & C' \cdot \Gamma' \text{ is a constrained clause over } \Sigma \text{ without} \\ & \text{expansion constraints, and} \\ & \Lambda^a \cup (\Phi \cup \{C \cdot \Gamma\})^c \models_{\mathrm{E}} (C' \cdot \Gamma')^c. \end{cases}$$

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

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 $D \cdot \Gamma$. Also, as observed previously, when the constraint Γ of a constrained clause $C \cdot \Gamma$ contains a rule $l \to r$ such that $l \prec r$ then this rule is trivially redundant wrt. any rewrite system ordered by $\succ$ and so the clause can be simplified to $\mathbf{t} \approx \mathbf{t} \cdot \emptyset$. 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.)

As illustrated by the last two examples, the practically important unit-resolution like rule of $\mathcal{ME}$, Resolve, is covered by the Simp rule.

Derivation Example. 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

[10] Except for the Subsume rule.

version $f(x) \approx x \vee f(f(x)) \approx b \vee f(b) \not\approx b$ of the ground clause there.

$$\ldots$$

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

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

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

(By Para)

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

(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.

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 [5].

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 $\mathbf{T}$).

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[11] 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 hard to see that actually carrying out a $\mathsf{Ref}_{\mathcal{ME}}$ 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 [3] accordingly.

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 not available during a derivation. The redundancy tests are therefore impossible to effectively realize in their full strength. Nethertheless, there are some effective and inexpensive redundancy tests similar to those discussed in conjunction with the Simp rule.

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 $\Box \cdot \Gamma$ that is productive wrt. $\Lambda_{\mathbf{B}}$ and that is not redundant wrt. $\Phi_{\mathbf{B}}$ and $R_{\Lambda_{\mathbf{B}}}$.*

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

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* **T** *be the limit tree of a fair derivation of Ψ. If* **T** *is not a refutation tree, then Ψ is satisfiable; more specifically, for every exhausted branch* **B** *of* **T**, $R_{\Lambda_{\mathbf{B}}} \models_{\mathrm{E}} \Psi$.

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 [3] 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

1. L. Bachmair and H. Ganzinger. Equational Reasoning in Saturation-Based Theorem Proving. In W. Bibel and P. H. Schmitt, ed., *Automated Deduction. A Basis for Applications*, Volume I: Foundations. Calculi and Refinements, pp. 353–398. Kluwer, 1998.
2. P. Baumgartner. FDPLL – A First-Order Davis-Putnam-Logeman-Loveland Procedure. In D. McAllester, ed., Proc. *CADE-17*, LNAI 1831, pp. 200–219. Springer, 2000.
3. P. Baumgartner, A. Fuchs, and C. Tinelli. Implementing the Model Evolution Calculus. *International Journal on Artificial Intelligence Tools (IJAIT)*, 2005. To appear.
4. P. Baumgartner and C. Tinelli. The Model Evolution Calculus. In F. Baader, ed., Proc. *CADE-19*, LNAI 2741, pp. 350–364. Springer, 2003.
5. P. Baumgartner and C. Tinelli. The Model Evolution Calculus with Equality, 2005. http://www.mpi-sb.mpg.de/~baumgart/publications/MEE.pdf.
6. J.-P. Billon. The Disconnection Method. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, eds., Proc. *TABLEAUX*, LNAI 1071, pp. 110–126. Springer, 1996.
7. M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem Proving. *Communications of the ACM*, 5(7):394–397, July 1962.
8. H. Ganzinger and K. Korovin. Integrating Equational Reasoning into Instantiation-Based Theorem Proving. In Proc. *CSL'04*, LNCS 3210, pp. 71–84. Springer, 2004.
9. H. Ganzinger and K. Korovin. New Directions in Instance-Based Theorem Proving. In Proc. *LICS*, 2003.
10. R. Letz and G. Stenz. Integration of Equality Reasoning into the Disconnection Calculus. In U. Egly and C. G. Fermüller, eds., *TABLEAUX*, LNCS 2381, pp. 176–190. Springer, 2002.
11. R. Nieuwenhuis and A. Rubio. Paramodulation-Based Theorem Proving. In J. A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*, pp. 371–443. Elsevier, 2001.
12. D. A. Plaisted and Y. Zhu. Ordered Semantic Hyper Linking. *Journal of Automated Reasoning*, 25(3):167–217, 2000.