Model Evolution with Equality Modulo Built-in Theories

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

Cesare Tinelli
The University of Iowa, USA,
cesare-tinelli@uiowa.edu

March 6, 2011

Abstract

Many applications of automated deduction require reasoning modulo background theories, in particular some form of integer arithmetic. Developing corresponding automated reasoning systems that are also able to deal with quantified formulas has recently been an active area of research. We contribute to this line of research and propose a novel instantiation-based method for a large fragment of first-order logic with equality modulo a given complete background theory, such as linear integer arithmetic. The new calculus is an extension of the Model Evolution Calculus with Equality, a first-order logic version of the propositional DPLL procedure, including its ordering-based redundancy criteria. We present a basic version of the calculus and prove it sound and (refutationally) complete under certain conditions.

1 Introduction

Many applications of automated deduction require reasoning modulo background theories, in particular some form of integer arithmetic. Older theory reasoning techniques developed within first-order theorem proving are often impractical as they require the enumeration of complete sets of theory unifiers (in particular those in the tradition of Stickel's Theory Resolution [14]) or feature only weak or no redundancy criteria (e.g., Bürckert's Constraint Resolution [7]). Developing sophisticated automated reasoning systems that are also able to deal with quantified formulas has recently been an active area of research [8, 10, 13, 4, 1]. We contribute to this line of research and propose a novel instantiation-based method for a large fragment of first-order logic with equality modulo a given complete background theory, such as linear integer arithmetic. The new calculus, $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$, is an extension of the Model

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

Evolution calculus with equality [5], a first-order logic version of the propositional DPLL procedure, including its ordering-based redundancy criteria as recently developed in [6]. At the same time, $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ is a generalization wrt. these features of the earlier $\mathcal{ME}(\mathrm{LIA})$ calculus [4].

Because instantiation based methods, including Model Evolution, have been shown to provide successful classical automated theorem proving methods we believe there is justification to develop theory-reasoning versions, even if the input logic or the decidability results are not new. Yet, we think $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ is relevant through its combination of powerful techniques for first-order equational logic with equality, based on an adaptation of the Bachmair-Ganzinger theory of superposition, with a black-box theory reasoner, which greatly enhances its versatility. $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ is thus similarly positioned as the hierarchic superposition calculus [1, 3].

Another angle to look at $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ is from SMT-solving: Over the last years, *Satisfiability Modulo Theories* has become a major paradigm for theorem proving modulo background theories. In one of its main approaches, DPLL(T), a DPLL-style SAT-solver is combined with a decision procedure for the quantifier-free fragment of the background theory T [11]. DPLL(T) is essentially limited to the ground case and resorts to incomplete or inefficient heuristics to deal with quantified formulas [9, e.g.]. In fact, addressing this intrinsic limitation by lifting DPLL(T) to the first-order level is one of the main motivations for the $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ calculus (much like Model Evolution was motivated by the goal of lifting the propositional DPLL procedure to the first-order level while preserving its good properties).

One possible application of $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ is for finite model reasoning. For example, the three formulas $1 \leq a \leq 100$, $P(a)$ and $\neg P(x) \leftarrow 1 \leq x \wedge x \leq 100$ together are unsatisfiable because the interval declaration $1 \leq a \leq 100$ for the constant a together with the unit clause $P(a)$ permit only models that satisfy one of $P(1), \ldots, P(100)$. Such models however falsify the third formula. Finite model finders, e.g., need about 100 steps to refute the clause set, one for each possible value of a. Our $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ calculus, on the other hand, can reason directly with integer intervals and allows a refutation in $O(1)$ steps. See Section 7 for further discussion of how this is achieved, variatons of the example, and considerations on $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ as a decision procedure.

The perhaps most promising application area is within software verification. Quite frequently, proof obligations arise that require quantified formulas to define data structures with *specific* properties, e.g., ordered lists or ordered arrays, and to prove that these properties are preserved under certain operations, e.g., when an element is inserted at an appropriate position. In the array case, one could define ordered arrays via "for all i, j with $0 \leq i < j \leq m$ it holds $a[i] < a[j]$" , where i and j are variables and m is a parameter, all integer-valued. Our calculus natively supports parameters like m and is well suitable for reasoning with integer variables that are bounded, like i and j. In general, parameters like m must be additionally constrained to a finite domain for the calculus to be effective, see again Section 7.

The general idea behind our calculus wrt. theory reasoning is to use ("rigid") variables to represent individual, yet at the current time unknown background domain elements, and instantiate these as needed to drive a derivation. As a simple example without parameters,

consider the clauses $f(x) \approx g(x) \leftarrow x > 5$ and $\neg(f(y+y) \approx g(8))$. These clauses will be refuted, essentially, by checking satisfiability of the set $\{v_1 = v_2 + v_2, v_1 > 5, v_1 = 8\}$ of constraints over rigid variables and (ordered) paramodulation inferences for reasoning with the equations in these clauses.

2 Preliminaries

We work in the context of standard many-sorted logic with first-order signatures comprised of sorts and operators (i.e., function symbols and predicate symbols) of given arities over these sorts. We rely on the usual notions of structure, (well-sorted) term/formula, satisfiability, and so on. If Σ is a sorted signature and X a set of sorted variables we will call *$\Sigma(X)$-term (resp. -formula)* a well-sorted term (resp. formula) built with symbols from Σ and variables from X. The notation $\Sigma(X_1, X_2)$ is a shorthand for $\Sigma(X_1 \cup X_2)$.

Syntax. For simplicity, we consider here only signatures with at most two sorts: a *background* sort B and a *foreground* sort F. We assume a *background signature* Σ_B having B as the only sort and an at most countable set of operators that includes an (infix) equality predicate symbol $=$ of arity $\mathsf{B} \times \mathsf{B}$. We will write $s \neq t$ as an abbreviation of $\neg(s = t)$. We fix an infinite set X_B of *B-variables*, variables of sort B.

We assume a first-order *background theory* T of signature Σ_B all of whose models interpret $=$ as the identity relation. Since T is complete, with no loss of generality we specify it simply as a Σ_B-structure. We call the set $|\mathsf{B}|$ that T associates to the sort B the *background domain*. We assume, again with no loss of generality, that $|\mathsf{B}|$ is at most countably infinite and all of its elements are included in Σ as B-constant symbols.[1] Our running example for T will be the theory of linear integer arithmetic (LIA). For that example, Σ_B's operators are $\leq, +$ and all the integer constants, all with the expected arities, T is the structure of the integer numbers with those operators, and $|\mathsf{B}| = \{0, \pm 1, \pm 2, \ldots\}$.

We will consider formulas over an expanded signature Σ_B^Π and expanded set of variables $X_\mathsf{B} \cup V$ where Σ_B^Π is obtained by adding to Σ_B an infinite set Π of *parameters*, free constants of sort B, and V is a set of B-variables not in X_B, which we call *rigid variables*. When we say just "variable" we will always mean a variable in X, not a rigid variable. The function and predicate symbols of Σ_B^Π are collectively referred to as the *background operators*. We call *(background) constraint* any formula in the closure of the set of $\Sigma_\mathsf{B}^\Pi(X_\mathsf{B}, V)$-atoms under conjunction, negation and existential quantification of variables.[2] A *closed constraint* is a constraint with no free variables (but possibly with rigid variables).

Note that rigid variables always occur free in a constraint. We will always interpret distinct rigid variables in a constraint as distinct elements of $|\mathsf{B}|$. Intuitively, in the calculus presented here, a rigid variable v will stand for a specific, but unspecified, background

[1] The latter assumption can be relaxed to the assumption that $|\mathsf{B}|$ is *Σ-generated*, meaning that every background domain element $n \in |\mathsf{B}|$ is the interpretation of a ground Σ-term.

[2] It is apparent though from how the calculus works that we only need a decision procedure for the validity of the $\forall\exists$-fragment over the class of constraints that the background constraints in formulas are drawn from. In the LIA case, one can use quantifier elimination for that. Without parameters, the $\exists$-fragment is sufficient.

domain element, and will be introduced during proof search similarly to rigid variables in free-variable tableaux calculi. In contrast, parameters will be free constants in input formulas, standing for arbitrary domain values.

The *full signature* Σ for our calculus is obtained by adding to $\Sigma_{\mathsf{B}}^{\Pi}$ the foreground sort F, function symbols of given arities over B and F, and one infix equality predicate symbol, $\approx$, of arity $\mathsf{F} \times \mathsf{F}$. The new function symbols and $\approx$ are the *foreground operators*. As usual, we do not consider additional foreground predicate symbols because they can be encoded as function symbols, e.g., an atom of the form $P(t_1, \ldots, t_n)$ can be encoded as $P(t_1, \ldots, t_n) \approx \mathsf{tt}$, where tt is a new, otherwise unused, foreground constant. For convenience, however, in examples we will often write the former and mean the latter. Since $\approx$ will always denotes a congruence relation, we will identify every equational atom $s \approx t$ with $t \approx s$.

Let X_{F} be an infinite set of *F-variables*, variables of sort F, disjoint with X_{B}, and let $X = X_{\mathsf{B}} \cup X_{\mathsf{F}}$. The calculus takes as input $\Sigma(X)$-formulas of a specific form, defined later, and manipulate more generally $\Sigma(X, V)$ formulas, i.e., formulas possibly containing rigid variables. We use, possibly with subscripts, the letters $\{x, y\}, \{u, v\}$, $\{a, b\}$, and $\{f, e\}$ to denote respectively regular variables (those in X), rigid variables, parameters, and foreground function symbols.

To simplify the presentation here, *we restrict the return sort of all foreground function symbols to be* F . This is a true restriction for non-constant function symbols.[3] For example, if Σ is the signature of lists of integers, with T being again LIA and F being the list sort, our logic allows formulas like $cdr(cons(x, y)) \approx y$ but not $car(cons(x, y)) \approx x$, as *car* would be integer-sorted. To overcome this limitation somewhat, one could turn *car* into a predicate symbol and use $car(cons(x, y), x)$ instead, together with the (universal) functionality constraint $\neg car(x, y) \vee \neg car(x, z) \leftarrow y \not\approx z$.

A *term* is a (well-sorted) $\Sigma(X, V)$-term. A *formula* is a (well-sorted) $\Sigma(X, V)$-formula. A *foreground term* is a term with no operators from $\Sigma_{\mathsf{B}}^{\Pi}$. Foreground atoms, literals, and formulas are defined analogously. An *ordinary foreground clause* is a multiset of foreground literals, usually written as a disjunction. A *background term* is a (well-sorted) $\Sigma_{\mathsf{B}}^{\Pi}(X_{\mathsf{B}}, V)$-term. Note that background terms are always B-sorted and vice versa. Foreground terms are made of foreground symbols, variables and rigid variables; they are all F-sorted unless they are rigid variables. A *ground term* is a term with no variables and no rigid variables. A *Herbrand term* is a ground term whose only background subterms are background domain elements. Intuitively, Herbrand terms do not contain symbols that need external evaluation, i.e., they contain no parameters, no variables, and no rigid variables. For example, $f(e, 1)$ and 1 are Herbrand terms, but $f(v, 1)$ and $f(a, 1)$ are not.

In this paper, a *substitution* will be any mapping σ from variables (in X) to terms that is sort respecting, that is, maps each variable x to a term of the same sort as x. We write substitution application in postfix form and extend the notation to (multi)sets S of terms or formulas in the obvious way, that is, $S\sigma = \{F\sigma \mid F \in S\}$. The *domain* of a substitution σ is the set $\mathrm{dom}(\sigma) = \{x \mid x \neq x\sigma\}$. A *Herbrand substitution* is a substitution that maps every variable to a Herbrand term. We denote by *fvar(F)* the set of non-rigid variables that occur

[3] Foreground constant symbols of sort B could be treated in the calculus just like parameters, and so they are not needed.

free in F, where F is a term or formula.

We assume a reduction ordering $\succ$ that is total on the Herbrand terms.[4] We also require that $\succ$ is stable under assignments, i.e., if $s \succ t$ then $s\alpha \succ t\alpha$, for every suitable assignment α for s and t. The ordering $\succ$ is extended to literals over Herbrand terms by identifying a positive literal $s \approx t$ with the multiset $\{s, t\}$, a negative literal $\neg(s \approx t)$ with the multiset $\{s, s, t, t\}$, and using the multiset extension of $\succ$. Multisets of literals are compared by the multiset extension of that ordering, also denoted by $\succ$.

Semantics. An *interpretation* I is any Σ-structure augmented to include an injective, possibly partial, mapping from the set V of rigid variables to the domain of B in I. We will be interested primarily in Herbrand interpretations, defined below.

Definition 2.1 (Herbrand interpretations) A *(T-based) Herbrand interpretation* is any interpretation I that (i) is identical to T over the symbols of Σ^{B}, (ii) interprets every foreground n-ary function symbol f as itself, i.e., $f^I(d_1, \ldots, d_n) = f(d_1, \ldots, d_n)$ for every tuple $(d_1, \ldots, d_n)$ of domain elements from the proper domain, and (iii) interprets $\approx$ as a congruence relation on F-sorted Herbrand terms.[5] □

A *(parameter) valuation* π is a mapping from Π to $|\mathsf{B}|$. An *assignment* α is an *injective* mapping from a (finite or infinite) subset of V to $|\mathsf{B}|$. Since T is fixed, a Herbrand interpretation I is completely characterized by three components: a congruence relation on the Herbrand terms, a valuation π and an assignment α.

An assignment α is *suitable* for a formula or set of formulas F if its domain includes all the rigid variables occurring in F. Since all the elements of $|\mathsf{B}|$ are constants of Σ_{B} we will often treat assignments and valuations similarly to substitutions. For any Herbrand interpretation I, valuation π and assignment α, we denote by $I[\pi]$ the interpretation that agrees with π on the meaning of the parameters (that is, $a^I = a\pi$ for all $a \in \Pi$) and is otherwise identical to I; we denote by $I[\alpha]$ the interpretation that agrees with α on the meaning of the rigid variables in α's domain and is otherwise identical to I. We write $I[\pi, \alpha]$ as a shorthand for $I[\pi][\alpha]$.

The symbols I, α and π we will always denote respectively Herbrand interpretations, assignments and valuations. Hence, we will often use the symbols directly, without further qualification. We will do the same for other selected symbols introduced later. Also, we will often implicitly assume that α is suitable for the formulas in its context.

Definition 2.2 (Satisfaction of constraints) Let c be a closed constraint. For all π and all α suitable for c, the pair (π, α) *satisfies* c, written as $(\pi, \alpha) \models c$, if $T \models c\pi\alpha$ in the standard sense.[6] If α is suitable for a set Γ of closed constraints, (π, α) *satisfies* Γ, written $(\pi, \alpha) \models \Gamma$, iff (π, α) satisfies every $c \in \Gamma$. □

[4] A *reduction ordering* is a strict, well-founded ordering on terms that is compatible with contexts, i.e., $s \succ t$ implies $f[s] \succ f[t]$, and stable under substitutions, i.e., $s \succ t$ implies $s\sigma \succ t\sigma$.

[5] Note that Condition (iii) is well defined because, by Condition (ii), the interpretation of the sort F is the set of all F-sorted Herbrand terms.

[6] Observe that the test $T \models c\pi\alpha$ is well formed because $c\pi\alpha$ is closed and contains neither parameters nor rigid variables.

The set Γ above is *satisfiable* if $(\pi, \alpha) \models \Gamma$, for some π and α. Since constraints contain no foreground symbols, for any interpretation $I[\pi, \alpha]$, $I[\pi, \alpha] \models c$ iff $(\pi, \alpha) \models c$.

Note 2.3 (Deciding the satisfiability of closed constraints) The satisfiability of arbitrary closed constraints, which may contain rigid variables, reduces in a straightforward way to the satisfiability of Σ_B-constraints without rigid variables, and so can be decided by any decision procedure for the latter. In fact, let V_Γ be the set of all the rigid variables occurring in a set Γ of closed constraints. Then, Γ is satisfiable iff $\mathrm{Distinct}(V_\Gamma) \cup \Gamma$ is satisfiable in T in the usual sense, where all the rigid variables and the parameters of Γ are treated as free variables, and $\mathrm{Distinct}(U) = \{v \neq u \mid v \in U, u \in U \setminus v\}$ for all $U \subseteq V$. The constraint set $\mathrm{Distinct}(V_\Gamma)$ reflects the injectivity of assignments.

This method obviously applies also to *finite* sets of closed constraints by taking conjunctions. For example, $\{3 > v_1, d_v > 1, v_2 = 2\}$ is not satisfiable (in LIA), because $v_1 \neq v_2 \wedge 3 > v_1 \wedge v_1 > 1 \wedge v_2 = 2$ is not satisfiable. □

3 Contexts and Constrained clauses

The calculus maintains two data structures for representing Herbrand interpretations: a *foreground context*, for the foreground operators, and a *background context*, for valuations and assignments. The former is a finite set of foreground literals, which we call *context literals*. The latter is a finite set of closed constraints. A *context* is a pair $\Lambda \cdot \Gamma$ consisting of a foreground context Λ and a background context Γ.

The symbols Λ and Γ will always denote respectively foreground contexts and background contexts, even without further qualification. We identify every foreground context Λ with its closure under renaming of (regular) variables, and assume it contains a pseudo-literal of the form $\neg x$. A foreground literal K is *contradictory with* Λ if $\overline{K} \in \Lambda$, where $\overline{K}$ denotes the complement of K. Λ itself is *contradictory* if it contains a literal that is contradictory with Λ. We will work only with non-contradictory contexts.

For any foreground literals K and L, we write $K \gtrsim L$ iff L is an instance of K, i.e., iff there is a substitution σ such that $K\sigma = L$. We write $K \sim L$ iff K and L are variants, equivalently, iff $K \gtrsim L$ and $L \gtrsim K$. We write $K \gtrsim L$ iff $K \gtrsim L$ but $L \not\gtrsim K$.

Definition 3.1 (Productivity) Let K, L be foreground literals. We say that *K produces L in Λ* if (i) $K \gtrsim L$, and (ii) there is no $K' \in \Lambda$ such that $K \gtrsim \overline{K'} \gtrsim L$. □

Since foreground contexts contain the pseudo-literal $\neg x$, it is not difficult to see that Λ produces either K or $\overline{K}$, for every non-contradictory Λ and literal K.

Definition 3.2 (Context unifier) Let $C = L_1 \vee \cdots \vee L_n$ be an ordinary foreground clause with $n \geq 0$. A substitution σ is a *context unifier of C against* Λ if there are literals $K_1, \ldots, K_n \in \Lambda$ such that σ is a simultaneous most general unifier of the sets $\{K_1, \overline{L}_1\}, \ldots, \{K_n, \overline{L}_n\}$. The literals $K_1, \ldots, K_n$ are the *context literals of σ*. □

We say that σ is *productive* iff K_i produces $\overline{L_i}\sigma$ in Λ, for all $i = 1, \ldots, n$.

Observe that context unifiers treat rigid variables like constants. A context unifier σ can be computed by composing most general unifiers of the literals in C with fresh variants of literals in Λ, one after another, while ignoring the sorts. That σ will be sort-respecting, and hence a substitution in our sense, which follows from the well-sortedness of literals and clauses.

For, if x is a background variable then $x\sigma$ can only be another background variable or rigid variable (these are the only B-sorted terms in foreground formulas); and if x is a foreground variable, $x\sigma$ can only be a foreground term occurring in C, possibly further instantiated in the same way.

The calculus works with *constrained clauses*, expressions of the form $C \leftarrow R \cdot c$ where R is a multiset of foreground literals, the set of *context restrictions*, C is an ordinary foreground clause, and c is a (background) constraint with $fvar(c) \subseteq fvar(C) \cup fvar(R)$. When C is empty we write it as $\square$. When R is empty, we write the constraint clause more simply as $C \leftarrow c$. The calculus takes as input only clauses of the latter form, hence we call such clauses *input constrained clauses*. Below we will often speak of (input) clauses instead of (input) constrained clauses when no confusion can arise.

We can turn any expression of the form $C \leftarrow c$ where C is an arbitrary ordinary Σ-clause and c a constraint into an input clause by abstracting out offending subterms from C, moving them to the constraint side of $\leftarrow$, and existentially quantifying variables in the constraint side that do not occur in the clause side. For example, $P(a, v, x + 5) \leftarrow x > v$ becomes $P(x_1, v, x_2) \leftarrow \exists x\, (x > v \wedge x_1 = a \wedge x_2 = x + 5)$. As will be clear later, this transformation preserves the semantics of the original expression.

The variables of input clauses are implicitly universally quantified. Because the background domain elements (such as, e.g., $0, 1, -1, \ldots$) are also background constants, we can define the semantics of input clauses in terms of Herbrand interpretations. To do that, we need one auxiliary definitions first.

If γ is a Herbrand substitution and $C \leftarrow c$ an input clause, the clause $(C \leftarrow c)\gamma = C\gamma \leftarrow c\gamma$ is a *Herbrand instance* of $C \leftarrow c$. For example, $(P(v, x, y) \leftarrow x > a)\gamma$ is $P(v, 1, f(1, e)) \leftarrow 1 > a$ if $\gamma = \{x \mapsto 1, y \mapsto f(1, e), \ldots\}$. A Herbrand instance $C \leftarrow c$ can be evaluated directly by an interpretation $I[\alpha]$, for suitable α: we say that $I[\alpha]$ *satisfies* $C \leftarrow c$, written $I[\alpha] \models C \leftarrow c$ if $I[\alpha] \models C \vee \neg c$. For input clauses $C \leftarrow c$ we say that $I[\alpha]$ *satisfies* $C \leftarrow c$ iff $I[\alpha]$ satisfies every Herbrand instance of $C \leftarrow c$.

Definition 3.3 (Satisfaction of sets of formulas) Let Δ be a set of input clauses and closed constraints. We say that $I[\alpha]$ *satisfies* Δ, written as $I[\alpha] \models \Delta$, if $I[\alpha] \models F$, for every $F \in \Delta$. $\square$

We say that Δ *is satisfiable* if some $I[\alpha]$ satisfies F. Let G be an input clause or closed constraint. We say that Δ *entails* G, written as $\Delta \models G$, if for every suitable assignment α for Δ and G, every interpretation $I[\alpha]$ that satisfies Δ also satisfies G.

The definition of satisfaction of general constraint clauses $C \leftarrow R \cdot c$, with a non-empty restriction R, is more complex because in our completeness argument for the calculus C is evaluated "semantically", with respect to Herbrand interpretations induced by a context, whereas R is evaluated "syntactically", with respect to productivity in a context. Moreover,

certain things cannot be expressed purely at the ground level and require to consider *Herbrand closures*.

Definition 3.4 (Herbrand closure) Let γ be a Herbrand substitution. The pair $(C \leftarrow R \cdot c, \gamma)$ is a *Herbrand closure (of $C \leftarrow R \cdot c$)*. □

Context restrictions are evaluated in terms of productivity by applying an assignment to the involved rigid variables first. To this end, we will use *evaluated contexts* $\Lambda\alpha = \{K\alpha \mid K \in \Lambda\}$. By the injectivity of α, the notions above on contexts apply "isomorphically" after evaluation by α. For instance, K produces L in Λ iff $K\alpha$ produces $L\alpha$ in $\Lambda\alpha$.

A set F of literals is *non-trivial* if no $K \in F$ is of the form $t \approx t$ or $\neg(t \approx t)$.

Definition 3.5 (Satisfaction of context restrictions) Let R be a set of context restrictions and γ a Herbrand substitution. The pair (Λ, α) *satisfies* (R, γ), written as $(\Lambda, \alpha) \models (R, \gamma)$, if

(i) $R\alpha\gamma$ is non-trivial, and for every $l \approx r \in R\alpha\gamma$, if $l \succ r$ then l is not a variable, and

(ii) for every $K \in R\alpha$ there is an $L \in \Lambda\alpha$ that produces both K and $K\gamma$ in $\Lambda\alpha$.

□

If Point (ii) above holds for some $K \in R\alpha$, we also say that $\Lambda\alpha$ produces K and $K\gamma$ by the same literal. Point (i) is need to avoid trivial equations and to show in the completeness proof why paramodulation into variables is not necessary.

Definition 3.6 (Satisfaction of Herbrand closures) A triple (Λ, α, I) *satisfies* $(C \leftarrow R \cdot c, \gamma)$, written as $(\Lambda, \alpha, I) \models (C \leftarrow R \cdot c, \gamma)$, iff $(\Lambda, \alpha) \not\models (R, \gamma)$ or $I \models (C \leftarrow c)\gamma$. □

We will use Definition 3.6 always with $I = I[\alpha]$. The component Λ in the previous definition is irrelevant for input clauses (where $R = \emptyset$), and satisfaction of Herbrand closures and Herbrand instances coincide then. Formally, $(\Lambda, \alpha, I[\alpha]) \models (C \leftarrow \emptyset \cdot c, \gamma)$ if and only if $I[\alpha] \models (C \leftarrow c)\gamma$.

In our soundness arguments for the calculus a constraint clause $C \leftarrow R \cdot c$ will stand for the Σ-formula $C \vee (\bigvee_{L \in R} \overline{L}) \vee \neg c$. We call the latter the *clause form* of $C \leftarrow R \cdot c$ and denote it by $(C \leftarrow R \cdot c)^{\mathrm{c}}$. If Φ is a set of clauses, $\Phi^{\mathrm{c}} = \{F^{\mathrm{c}} \mid F \in \Phi\}$.

4 Core Inference Rules

The calculus works on sequents of the form $\Lambda \cdot \Gamma \vdash \Phi$, where $\Lambda \cdot \Gamma$ is a context and Φ is a set of constrained clauses, all components finite. We write $\Lambda, K \cdot \Gamma \vdash \Phi$ for $\Lambda \cup \{K\} \cdot \Gamma \vdash \Phi$ and similarly with Γ and Φ. There are five core inference rules: Pos-Res, Ref, Para, Split and Close.

The first two inference rules perform equality reasoning at the foreground level.

$$\mathsf{Ref}\ \frac{\Lambda \cdot \Gamma \vdash \Phi}{\Lambda \cdot \Gamma \vdash \Phi, (C \leftarrow R \cdot c)\sigma}$$

if Φ contains a clause $\neg(s \approx t) \vee C \leftarrow R \cdot c$ and σ is an mgu of s and t. The clause $\neg(s \approx t) \vee C \leftarrow R \cdot c$ is the *selected clause*, and the clause in the conclusion is the *derived clause*.

The next inference rule is a variant of ordered paramodulation.

$$\textsf{Para}\ \frac{\Lambda \cdot \Gamma \vdash \Phi}{\Lambda \cdot \Gamma \vdash \Phi, (L[r] \vee C \leftarrow (R \cup \{l \approx r\}) \cdot c)\sigma}$$

if $l \approx r \in \Lambda$ and Φ contains a clause $L[s] \vee C \leftarrow R \cdot c$ such that (i) σ is an mgu of l and s, (ii) s is not a variable and s is not a rigid variable, (iii) $r\sigma \not\succeq l\sigma$, and (iv) $l \approx r$ produces $(l \approx r)\sigma$ in Λ.

The clause $L \vee C \leftarrow R \cdot c$ is the *selected clause*, the context literal $l \approx r$ is the *selected context equation*, and the new clause in the conclusion is the *derived clause*.

We can afford to not paramodulate into rigid variables s, as these are B-sorted, and the resulting unifier with (a F-sorted variable) l would be ill-sorted. The equation $l \approx r$ is added to R to preserve soundness.

Example 4.1 Let $\Lambda = \{f(x,y,e) \approx x\}$. Then the clause $P(f(x,e,y)) \vee y \approx e \leftarrow \emptyset \cdot x > 5$ paramodulates into $P(x) \vee e \approx e \leftarrow f(x,e,e) \approx x \cdot x > 5$. □

If L is a negative equation $t_1[s] \not\approx t_2$, the Para rule could be improved by requiring that $t_2\sigma \not\succeq t_1\sigma$. That is, paramodulation into smaller sides of negative equations is not necessary.

We could afford a *selection function* that select zero or more occurrences of negative equations in the component C of a clause $C \leftarrow R \cdot c$. The Ref and Para inference rules then must selected a literal that is among the ones selected by the selection function, if there are any. The rationale for this restriction is that negative equations must be "proven" by making their two sides (syntactically) equal anyway, so this can be done at any time.

If C is an ordinary foreground clause, $\overline{C}$ denotes the multiset of the complements of the literals in C, i.e. $\overline{C} = \{\overline{L} \mid L \in C\}$, which is a set of context restrictions.

The next rule turns the ordinary clause part of a constrained clause into context restrictions.

$$\textsf{Pos-Res}\ \frac{\Lambda \cdot \Gamma \vdash \Phi}{\Lambda \cdot \Gamma \vdash \Phi, (\Box \leftarrow (R \cup \overline{C}) \cdot c)\sigma}$$

if Φ contains a clause of the form $C \leftarrow R \cdot c$ such that (i) $C \neq \Box$ and C consists of positive literals only, and (ii) σ is a productive context unifier of C against Λ. The clause $C \leftarrow R \cdot c$ is the *selected clause*, and the new clause in the conclusion is the *derived clause*.

Note that the derived clause is indeed a constraint clause. The reason is that the literals of C and the context literals used for the unifier σ are all foreground literals. As a consequence, σ replaces foreground variables by foreground terms and background variables by background variables or rigid variables. For example, if $\Lambda = \{\neg P(e)\}$, from $f(x,y,z) \approx g(y) \vee P(x) \leftarrow \emptyset \cdot y > 5$ one gets $\Box \leftarrow \{\neg(f(e,y,z) \approx g(y)), \neg P(e)\} \leftarrow \emptyset \cdot y > 5$ (recall that implicitly $\neg v \in \Lambda$).

Intuitively, Pos-Res is applied when all literals in the ordinary clause part of a clause have been sufficiently processed by the equality inference rules Para and Ref (including the positive ones) and turns them into context restriction. Deriving an empty constrained clause this way does not make a refutation, though, as the clause could possibly be satisfied, in an interpretation that falsifies its context restriction or falsifies its constraint. The Split rule below analysis this possibility. It takes a constrained empty clause from Φ and splits on one of its context restriction literals, after instantiating all free variables in the constraint (only) by some rigid variables.

It comes with side conditions that treat context literals as constrained clauses. Formally, let $\Lambda^{(e,n)} = \{K^{(e,n)} \leftarrow \top \mid K \in \Lambda\}$ be the *clause form of* Λ, where $K^{(e,n)}$ is the context literal obtained from K by replacing every foreground variable by a fixed foreground constant e and replacing every background variable by a fixed background domain element n. We say that $(C \leftarrow R{\cdot}c)\delta$ is a *domain instance* of a clause $C \leftarrow R{\cdot}c$ if δ moves every B-sorted variable of $\mathit{fvar}(c)$ to a rigid variable and does not move the other variables of $\mathit{fvar}(c)$.

$$\mathsf{Split}\ \frac{\Lambda \cdot \Gamma \vdash \Phi}{\Lambda, \overline{K} \cdot \Gamma, c \vdash \Phi \qquad \Lambda, K \cdot \Gamma^\star \vdash \Phi}$$

if there is a domain instance $(\Box \leftarrow R \cdot c)$ of some clause in Φ such that (i) $K \in R$ and neither $\overline{K}$ nor K is contradictory with Λ, (ii) for every $L \in R$, Λ produces L, (iii) $\Gamma \cup \{c\}$ is satisfiable, and (iv) $\Gamma^\star$ is any satisfiable background context such that $\Gamma \cup \{c\} \subseteq \Gamma^\star$ and $(\Lambda \cup \overline{K})^{(e,n)} \cup \Phi^{\mathrm{c}} \cup \Gamma^\star$ is not satisfiable, if such a $\Gamma^\star$ exists, or else $\Gamma^\star = \Gamma \cup \{c\}$. The clause $\Box \leftarrow R \cdot c$ is the *selected clause*, and the literal $\overline{K}$ is the *split literal*.

For example, if $\Lambda = \{\neg P(e)\}$ and Φ contains $\Box \leftarrow \{\neg(f(e,y,z) \approx g(y)), \neg P(e) \cdot y > 5$, the domain instance could be $\Box \leftarrow \{\neg(f(e,v_1,z) \approx g(v_1)), \neg P(e)\} \cdot v_1 > 5$, and the split literal can be (only) $f(e,v_1,z) \approx g(v_1)$.

The set Φ can also be seen to implicitly contain with each clause all its domain instances, and taking one of those as the selected clause for Split. For soundness reasons, background contexts need to be global to derivations. Any conditions added to Γ in the course of a further derivation of the left branch need to be present in the right branch. (See the proof of Theorem 7.1, soundness.) This is modeled by Condition (iv). The branch $\Gamma^\star$ can be obtained in a constructive way by trying to extend the left branch to a refutation sub-tree, which, if successful, gives the desired $\Gamma^\star$. If not successful, no matter if finite or infinite, the input clause set is satisfiable, and the derivation need not return to the right branch anyway.

$$\mathsf{Close}\ \frac{\Lambda \cdot \Gamma \vdash \Phi}{\Lambda \cdot \Gamma, c \vdash \Phi, (\Box \leftarrow \emptyset \cdot \top)}$$

if Φ contains a clause $\Box \leftarrow R{\cdot}c$ such that (i) $R \subseteq \Lambda$, and (ii) $\Gamma \cup \{c\}$ is satisfiable. The clause $\Box \leftarrow R \cdot c$ is the *selected clause*. Observe that if Split is applicable then Close cannot be applied with the same selected clause, as its Condition (i) will not be satisfied.

5 Model Construction, Redundancy and Static Completeness

In this section we will show how derive from a sequent $\Lambda \cdot \Gamma \vdash \Phi$ an intended interpretation $I[\Lambda, \pi, \alpha]$ as a canonical candidate model for Φ. Its components π and α will be determined first by Γ, and its congruence relation will be presented by a convergent ground rewrite system $R_{\Lambda,\alpha}$ extracted from Λ and α. The general technique for defining $R_{\Lambda,\alpha}$ is borrowed from the completeness proof of the Superposition calculus [2, 12] and earlier $\mathcal{ME}$ calculi [6, 5]. One difference is that $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ requires the construction of a fully reduced rewrite system, whereas for Superposition a left-reduced rewrite system is sufficient.

A *rewrite rule* is an expression of the form $l \to r$ where l and r are F-sorted Herbrand terms. A *rewrite system* is a set of rewrite rules. The rewrite systems constructed below will be ordered, that is, consist of rules of the form $l \to r$ such that $l \succ r$. For a given Λ and suitable assignment α, we define by induction on the term ordering $\succ$ sets ϵ_K and R_K for every ground equation K between F-sorted Herbrand-terms. Assume that ϵ_L has already been defined for all such L with $K \succ L$. Let $R_K = \bigcup_{K \succ L} \epsilon_L$, where

$$\epsilon_{l \approx r} = \begin{cases} \{l \to r\} & \text{if } \Lambda\alpha \text{ produces } l \approx r,\ l \succ r, \text{ and } l \text{ and } r \text{ are irreducible} \\ & \text{wrt } R_{l \approx r} \\ \emptyset & \text{otherwise} \end{cases}$$

Finally define $R_{\Lambda,\alpha} = \bigcup_K \epsilon_K$. If $\epsilon_{l \approx r} = l \to r$ we say that *$l \approx r$ generates $l \to r$ in $R_{\Lambda,\alpha}$*.

For example, if $\Lambda = \{P(x), \neg P(v)\}$ and $\alpha = \{v \mapsto 1\}$ then $R_{\Lambda,\alpha}$ contains $P(0)$, $P(-1)$, $P(-2)$, $P(2)$, $P(-3)$, $P(3), \ldots$ but not $P(1)$, which although irreducible, is not produced by $\Lambda\alpha$.

Definition 5.1 (Induced interpretation) Let Λ be a context, π be a valuation, and α a suitable assignment for Λ. The *interpretation induced by Λ, π and α*, written as $I[\Lambda, \pi, \alpha]$, is the Herbrand interpretation $I[\pi, \alpha]$ that interprets foreground equality as $R^{\star}_{\Lambda,\alpha}$, i.e., the smallest congruence relation on Herbrand terms. □

The rewrite system $R_{\Lambda,\alpha}$ is fully reduced by construction (no rule in $R_{\Lambda,\alpha}$ rewrites any other rule in it). Since $\succ$ is well-founded on the Herbrand terms, $R_{\Lambda,\alpha}$ is convergent. It follows with well-known results that equality of Herbrand terms in $R^{\star}_{\Lambda,\alpha}$ can be decided by reduction to normal form using the rules in $R_{\Lambda,\alpha}$.

The rewrite system $R_{\Lambda,\alpha}$ will also be used to evaluate evaluated context restrictions:

Definition 5.2 (Satisfaction of variable-free foreground literals) Let R be a set of literals over Herbrand terms. We say that *$R_{\Lambda,\alpha}$ satisfies R*, and write $R_{\Lambda,\alpha} \models R$, iff

(i) for every $l \approx r \in R$, if $l \succ r$ then $l \to r \in R_{\Lambda,\alpha}$, and

(ii) for every $\neg(l \approx r) \in R$, l and r are irreducible wrt. $R_{\Lambda,\alpha}$.

□

For example, if $\Lambda = \{f(v) \approx e_2\}$, $\alpha = \{v \mapsto 1\}$, and $f(1) \succ e_1 \succ e_2 \succ 1$ then $R_{\Lambda,\alpha} = \{f(1) \to e_2\}$ and $R_{\Lambda,\alpha} \not\models \{\neg(f(1) \approx e_1), e_2 \approx e_1\}$ because the left-hand side of $\neg(f(1) \approx e_1)$ is reducible wrt. $R_{\Lambda,\alpha}$, and because $e_1 \to e_2$ is not in $R_{\Lambda,\alpha}$.

Our concepts of redundancy require comparing Herbrand closures. To this end, define $(C_1 \leftarrow R_1 \cdot c_1, \gamma_1) \succ (C_2 \leftarrow R_2 \cdot c_2, \gamma_2)$ iff $C_1\gamma_1 \succ C_2\gamma_2$, or else $C_1\gamma_1 = C_2\gamma_2$ and $R_1\gamma_1 \succ R_2\gamma_2$. Notice that despite constraints are ignored, this ordering is not total, as constrained clauses may contain rigid variables.

Definition 5.3 (Redundant clause) Let $\Lambda \cdot \Gamma \vdash \Phi$ be a sequent, and $\mathscr{D}$ and $(C \leftarrow R \cdot c, \gamma)$ Herbrand closures. We say that $(C \leftarrow R \cdot c, \gamma)$ is *redundant wrt $\mathscr{D}$ and $\Lambda \cdot \Gamma \vdash \Phi$* iff (a) there is a $K \in R$ that is contradictory with Λ, (b) $\Gamma \cup \{c\gamma\}$ is not satisfiable, or (c) there exist Herbrand closures $(C_i \leftarrow R_i \cdot c_i, \gamma_i)$ of clauses in Φ, such that all of the following hold:

(i) for every $L \in R_i$ there is a $K \in R$ such that $L \sim K$ and $L\gamma_i = K\gamma$,

(ii) $\Gamma \cup \{c\gamma\} \models c_i\gamma_i$,

(iii) $\mathscr{D} \succ (C_i \leftarrow R_i \cdot c_i, \gamma_i)$, and

(iv) $\{C_1\gamma_1, \ldots, C_n\gamma_n\} \models C\gamma$.

□

We say that a Herbrand closure $(C \leftarrow R \cdot c, \gamma)$ is *redundant wrt $\Lambda \cdot \Gamma \vdash \Phi$* iff it is redundant wrt $(C \leftarrow R \cdot c, \gamma)$ and $\Lambda \cdot \Gamma \vdash \Phi$, and that a clause $C \leftarrow R \cdot c$ is *redundant wrt $\Lambda \cdot \Gamma \vdash \Phi$* iff every Herbrand closure of $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$.

If case (a) or (b) in the previous definition applies then $(C \leftarrow R \cdot c, \gamma)$ is trivially satisfied by $(\Lambda, \alpha, I[\alpha])$, for every suitable α that satisfies Γ and every $I[\alpha]$. Case (c) provides with (ii) and (iv) conditions under which $(C \leftarrow c)\gamma$ follows from the $(C_i \leftarrow c_i)\gamma_i$'s (in the sense of Definition 3.3). The context restrictions are taken into account by condition (i), which makes sure that evaluation of the pairs (R_i, γ_i) in terms of Definition 3.5 is the same as for (R, γ). In condition (iv), entailment $\models$ is meant as entailment in equational clause logic between sets of ordinary ground clauses and an ordinary ground clause.

Given a Pos-Res, Ref or Para inference with premise $\Lambda \cdot \Gamma \vdash \Phi$, selected clause $C \leftarrow R \cdot c$, selected context equation $l \approx r$ in case of Para, and a Herbrand substitution γ. If applying γ to $C \leftarrow R \cdot c$, the derived clause, and $l \approx r$ satisfies all applicability conditions of that inference rule, except $(C \leftarrow R \cdot c)\gamma \in \Phi$ and $(l \approx r)\gamma \in \Lambda$, we call the resulting ground inference a *ground instance via γ (of the given inference)*. This is not always the case, as, e.g., ordering constraints can become unsatisfiable after application of γ.

Definition 5.4 (Redundant inference) Let $\Lambda \cdot \Gamma \vdash \Phi$ and $\Lambda' \cdot \Gamma' \vdash \Phi'$ be sequents. An inference with premise $\Lambda \cdot \Gamma \vdash \Phi$ and selected clause $C \leftarrow R \cdot c$ is *redundant wrt $\Lambda' \cdot \Gamma' \vdash \Phi'$* iff for every Herbrand substitution γ, $(C \leftarrow R \cdot c, \gamma)$ is redundant wrt. $\Lambda' \cdot \Gamma' \vdash \Phi'$ or the following holds, depending on the inference rule applied:

Pos-Res, Ref, Para: Applying γ to that inference does not result in a ground instance via γ, or $(C' \leftarrow R' \cdot c', \gamma)$ is redundant wrt. $(C \leftarrow R \cdot c, \gamma)$ and $\Lambda' \cdot \Gamma' \vdash \Phi'$, where $C' \leftarrow R' \cdot c'$ is the derived clause of that inference.

Split ($C = \square$): (a) there is a literal $K \in R$ such that Λ' does not produce K or (b) the split literal is contradictory with Λ'.

Close ($C = \square$): $\square \leftarrow \emptyset \cdot \top \in \Phi'$.

□

Definition 5.5 (Saturated sequent) A sequent $\Lambda \cdot \Gamma \vdash \Phi$ is *saturated* iff every inference with a core inference rule and premise $\Lambda \cdot \Gamma \vdash \Phi$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$. □

We note that actually carrying out an inference makes it redundant wrt. the (all) conclusion(s), which already indicates that saturated sequents can be effectively computed.

Our first completeness result holds only for saturated sequents with respect to *relevant* closures. We say that a clause $(C \leftarrow R \cdot c, \gamma)$ is *relevant wrt. Λ and α* iff $R_{\Lambda,\alpha} \models R\alpha\gamma$. All Herbrand closures of input clauses are always relevant.

Theorem 5.6 (Static completeness) *Let $\Lambda \cdot \Gamma \vdash \Phi$ be a saturated sequent, π a valuation and α a suitable assignment for $\Lambda \cdot \Gamma \vdash \Phi$. If $(\pi, \alpha) \models \Gamma$, $\mathrm{ran}(\alpha) = |\mathsf{B}|$ and $(\square \leftarrow \emptyset \cdot \top) \notin \Phi$ then the induced interpretation $I[\Lambda, \pi, \alpha])$ satisfies all Herbrand closures of all clauses in Φ that are relevant wrt. Λ and α. Moreover, $I[\Lambda, \pi, \alpha] \models C \leftarrow c$, for every $C \leftarrow c \in \Phi$.*

The stronger statement $I[\Lambda, \pi, \alpha] \models \Phi$ does in general not follow, as $I[\Lambda, \pi, \alpha]$ possibly does not satisfy a *non-relevant* closure of a clause in Φ. An example is the sequent (with $e_1 \succ e_2$) $\Lambda \cdot \Gamma \vdash \Phi = P(x),\ e_1 \approx e_2,\ \neg P(e_2) \cdot \emptyset \vdash P(x) \leftarrow P(x) \cdot \top$. For any α, we get $R_{\Lambda,\alpha} = \{e_1 \to e_2\}$. By taking $\gamma = \{x \mapsto e_1\}$ observe that $(\Lambda, \alpha) \models (P(x), \gamma)$ but $R^\star_{\Lambda,\alpha} \not\models P(x)\gamma$, hence $I[\Lambda, \pi, \alpha] \not\models P(x) \leftarrow P(x) \cdot \top$. Deriving $\square \leftarrow \neg P(x), P(x) \cdot \top$ does not help to close. But notice that $R^\star_{\Lambda,\alpha} \not\models \{P(x)\gamma\}$, as $P(e_1) \notin R_{\Lambda,\alpha}$, and so $P(x) \leftarrow P(x) \cdot \top$ is not a *relevant* closure wrt. Λ and α, and so Theorem 5.6 is not violated.

Theorem 5.6 applies to a *statically* given sequent $\Lambda \cdot \Gamma \vdash \Phi$. The connection to the *dynamic* derivation process of the $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ calculus will be given later, and Theorem 5.6 will be essential then in proving the completeness of the $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ calculus.

6 The $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ Calculus

We are now turning to derivation processes for computing saturated sequents. First we introduce two more inference rules. The first one, Simp, is a generic simplification rule.

$$\text{Simp}\ \frac{\Lambda \cdot \Gamma \vdash \Phi, C \leftarrow R \cdot c}{\Lambda \cdot \Gamma \vdash \Phi, C' \leftarrow R' \cdot c'}$$

if (1) $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi, C' \leftarrow R' \cdot c'$, and (2) $\Gamma \cup \Lambda^{(e,n)} \cup (\Phi \cup \{C \leftarrow R \cdot c\})^{\mathrm{c}} \models (C' \leftarrow R' \cdot c')^{\mathrm{c}}$. The first condition is needed for completeness, and the second condition is needed for soundness.

For example, if Λ contains a ground literal K, then every constraint clause of the form $C \leftarrow (\{\overline{K}\} \cup R) \cdot c$ can be deleted, and every constraint clause of the form $C \leftarrow (\{K\} \cup R) \cdot c$ can be replaced by $C \leftarrow R \cdot c$. The Simp rule encompasses various additional forms of simplification of the literals in C based on rewriting and subsumption, see [6].

We only note here that additional (optional) inference rules can be defined for simplifying contexts.

$$\text{Restrict}\ \frac{\Lambda \cdot \Gamma \vdash \Phi}{\Lambda \cdot \Gamma, c \vdash \Phi}$$

if c is a closed constraint such that $\Gamma \cup \{c\}$ is satisfiable.

For example, by 10-fold application of restrict one can construct a background context $\{1 \leq v_1 \leq 10, \ldots, 1 \leq v_{10} \leq 10\}$ that represents the numbers $1, \ldots, 10$ in a "nondeterministic" way. The purpose of Restrict is to construct finitely committed branches, as formally introduced below.

We are now going to introduce derivations in a formal way. Let Ψ is a set of pure input clauses and Γ a satisfiable set of closed constraints. A *derivation from Ψ and Γ* is a sequence $((N_i, E_i))_{0 \leq i < \kappa}$ of trees of sequents (called derivation trees) with nodes N_i and edges E_i, such that $\mathbf{T}_0$ consists of the root-only tree whose sequent is $\neg v \cdot \Gamma \vdash \Psi$, and $\mathbf{T}_i$ is obtained by one single application of one of the core inference rules, Simp or Restrict to $\mathbf{T}_{i-1}$, for all $1 \leq i < \kappa$.

A *refutation* is a derivation that contains a *refutation tree*, that is, a derivation tree that contains in each leaf a sequent with $\square \leftarrow \emptyset \cdot \top$ in its clauses.

Every derivation determines a possibly infinite *limit tree* $\mathbf{T} = (\bigcup_{i<\kappa} N_i, \bigcup_{i<\kappa} E_i)$. In the following, let $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ be the sequent labelling the node i in the branch $\mathbf{B}$ with κ nodes of a limit tree $\mathbf{T}$, for all $i < \kappa$. Let

- $\Gamma_{\mathbf{B}} = \bigcup_{i<\kappa} \Gamma_i$ the *limit background context*,
- $\Lambda_{\mathbf{B}} = \bigcup_{i<\kappa} \Lambda_i$ be the *limit foreground context literals*, and
- $\Phi_{\mathbf{B}} = \bigcup_{i<\kappa} \bigcap_{i \leq j < \kappa} \Phi_i$ be the *persistent clauses*.

The tuple $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is called the *limit sequent (of $\mathbf{B}$)*. To prove a completeness result, derivations in $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ need to construct limit sequents with certain properties. One of these properties is expressed in terms of the following closure property:

Definition 6.1 (Exhausted branch) We say that $\mathbf{B}$ is *exhausted* iff for all $i < \kappa$, both of the following hold:

(i) every Pos-Res, Ref, Para, Split and Close inference with premise $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ and a persistent selected clause is redundant wrt. $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$ for some j with $i \leq j < \kappa$.

(ii) $(\Box \leftarrow \emptyset \cdot \top) \notin \Phi_i$.

□

While the above notion is similar to the one already used in $\mathcal{ME}_{\mathrm{E}}$, $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ has additional requirements on the limit background context $\Gamma_{\mathbf{B}}$, introduced next.

For any background domain element n and assignment α let $\mathrm{pre}_\alpha(n) = \{\nu \mid \alpha(\nu) = n\}$. Because assignments are injective, $\mathrm{pre}_\alpha(n)$ is either a singleton or the empty set (if α does not map any ν to n). Similarly, for any rigid variable ν let $\mathrm{img}_\alpha(\nu) = \{\alpha(\nu)\}$ if $\nu \in \mathrm{dom}(\alpha)$, and $\mathrm{img}_\alpha(\nu) = \emptyset$ otherwise. Analogously, for any parameter a and parameter valuation π let $\mathrm{img}_\pi(a) = \{\pi(a)\}$ (recall that parameter valuations are total).

Definition 6.2 (Finitely committed branch) We say that **B** is *finitely committed* iff (a) $\Gamma_{\mathbf{B}}$ is finite or (b) for all $i < \kappa$, there are π_i and α_i such that $(\pi_i, \alpha_i) \models \Gamma_i$, and

(i) $\bigcup_{i<\kappa} \bigcap_{i\le j<\kappa} \mathrm{ran}(\alpha_j) = |\mathsf{B}|$,

(ii) for every background domain element n, $\bigcup_{i<\kappa} \mathrm{pre}_{\alpha_i}(n)$ is finite,

(iii) for every rigid variable ν occuring in $\Gamma_{\mathbf{B}}$, $\bigcup_{i<\kappa} \mathrm{img}_{\alpha_i}(\nu)$ is finite, and

(iv) for every parameter a occuring in $\Gamma_{\mathbf{B}}$, $\bigcup_{i<\kappa} \mathrm{img}_{\alpha_i}(a)$ is finite.

□

The set in condition (i) consists of those background domain elements that are represented by some (not neccessarily the same) rigid variable from some point on forever. Condition (i) then says this must be the case for all background domain elements. Condition (ii) then says that only finitely many rigid variables can be used for that. Condition (iii) says, in other words, that no rigid variable occuring in $\Gamma_{\mathbf{B}}$ can be assigned infinitely many values as the context evolves. Similarly in condition (iv) for parameters.

The purpose of Definition 6.2 is to make sure that a valuation π and a suitable assignment α for Γ always exists, and moreover, that (π, α) satisfies $\Gamma_{\mathbf{B}}$:

Proposition 6.3 (Compactness of finitely committed branches) *If* **B** *is finitely committed then there is a* π *and an* α *such that* $\mathrm{ran}(\alpha) = |\mathsf{B}|$ *and* $(\pi, \alpha) \models \Gamma_{\mathbf{B}}$.

Proposition 6.3 states a compactness property. By construction, the sequence $(\Gamma_i)_{i<\kappa}$ is an non-decreasing chain wrt $\subseteq$ and every Γ_i is satisfiable. Proposition 6.3 then takes satisfiability to the limit $\Gamma_{\mathbf{B}}$.

One might wonder if the lemma is in contradiction to well-known results. Linear integer arithmetic, for instance, is not compact (let d be a rigid variable and take the set $\{d \neq n \mid n \text{ is an integer}\}$, every finite subset of which is satisfable). However, the lemma is about sets $\Gamma_{\mathbf{B}}$ with specific properties.

To see one of the issues that Proposition 6.3 addresses consider $\Gamma_i = \bigcup_{k\le i}\{\nu_1 > k\}$ then $\Gamma_{\mathbf{B}}$ is not satisfiable, although every finite subset is satisfiable. But on the other hand condition (iii) in Definition 6.2 is not satisfied.

We only note here with enough Restrict applications finitely committed limit branches can be constructed in a straightforward way if the input clause set is rigid variable-free, which is an unproblematic assumption, and if the input background constraints essentially confine each parameters to a finite domain. In the LIA case, for example, one could "slice" the integers in intervals of, say, 100 elements and enumerate with *Restrict* declarations like $1 \leq \nu_1 \leq 100, \ldots, 1 \leq \nu_{100} \leq 100$ before any rigid variable ν_i is used for the first time (in Split), and do that for all intervals. In certain cases it is possible to determine *a priori* that limit background contexts will be finite, and then Restrict is not required at all, see Section 7.

Definition 6.4 (Fairness) A derivation is fair iff it is a refutation or its limit tree has an exhausted and finitely committed branch. □

The following proposition is instrumental in proving completeness:

Proposition 6.5 (Exhausted branches are saturated) *If* $\mathbf{B}$ *is an exhausted branch of a limit tree of a fair derivation then* $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$ *is saturated.*

Theorem 6.6 (Completeness) *Let* Ψ *be a set of input clauses,* Γ *a satisfiable set of closed constraints and* $\mathbf{T}$ *be the limit tree of a fair derivation from* Ψ *and* Γ *that is not a refutation tree. Let* $\mathbf{B}$ *be any exhausted and finitely committed branch of* $\mathbf{T}$*, and* $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$ *the limit sequent of* $\mathbf{B}$*.*

Let π *be a valuation and* α *a suitable assignment for* $\Gamma_\mathbf{B}$ *with* $\mathrm{ran}(\alpha) = |\mathsf{B}|$ *such that* $(\pi, \alpha) \models \Gamma_\mathbf{B}$*, which exist by Proposition 6.3. Then, the induced interpretation* $\mathcal{I}[\Lambda_\mathbf{B}, \pi, \alpha]$ *satisfies all Herbrand closures of all clauses in* $\Phi_\mathbf{B}$ *that are relevant wrt.* $\Lambda_\mathbf{B}$ *and* α*. Moreover,* $\mathcal{I}[\Lambda_\mathbf{B}, \pi, \alpha] \models \Psi$*.*

7 Soundness and Special Cases

Theorem 7.1 (Relative refutational soundness) *Let* $\mathbf{D}$ *be a refutation from a set* Ψ *of input clauses and a satisfiable set* Γ *of closed constraints.*

If $\mathbf{T}$ *is a refutation tree in* $\mathbf{D}$*,* $\mathbf{B}$ *the rightmost branch in* $\mathbf{T}$*, and* $\Gamma_\mathbf{B}$ *the sequent in the leaf of* $\mathbf{B}$*, then* $\Gamma_\mathbf{B} \supseteq \Gamma$*,* $\Gamma_\mathbf{B}$ *is satisfiable, and* $\Gamma_\mathbf{B} \cup \Psi$ *is not satisfiable.*

Typically, the calculus is applied to rigid variable-free input clause sets Ψ. If additionally Ψ is parameter-free, then Ψ is satisfiable or not, independent of parameter valuations and assignments. Theorem 7.1 then reduces to the conclusion that Ψ alone is not satisfiable. However, typically Ψ will contain parameters. Again assuming that Ψ is rigid variable-free, Theorem 7.1 then allows us to conclude $\mathcal{I}[\pi] \not\models \Psi$ for every $\mathcal{I}[\pi]$ and (π, α) such that $(\pi, \alpha) \models \Gamma$.

For example, if $\Psi = \{P(x) \leftarrow x = a,\ \neg P(x) \leftarrow x = 5\}$ and $\Gamma = \{a > 2\}$ then there is a refutation with $\Gamma_\mathbf{B} = \{a > 2, a = 5\}$. Of course, $\Psi \cup \Gamma$ is satisfiable, cf. Definition 3.3. A soundness result can thus be not based on *single* refutations, and this is why we call the soundness result above "relative". To obtain the expected soundness result, we work

with *sequences* of refutations whose limit background contexts collectively cover the initially given Γ. In the example, the next derivation starts with $\Gamma' = \{a > 2, \neg(a = 5)\}$, which leads to a (finite) derivation that provides the expected model.

Let Ψ be the input clause set and Γ the (rigid variable-free) satisfiable input background constraints. Say that π *satisfies* $\exists\Gamma$ if $(\pi, \alpha) \models \Gamma$, for some suitable α. The intuition above leads to the following general procedure:

repeat
  let **D** be a derivation from Ψ and Γ
  if **D** is a refutation then
    let $\Gamma_{\mathbf{B}}$ be its limit background context
    if every π that satisfies $\exists\Gamma$ also satisfies $\exists\Gamma_{\mathbf{B}}$ then stop with "unsatisfiable"
    else let $\Gamma' \supset \Gamma$ such that some π satisfies $\exists\Gamma'$ but does not satisfy $\Gamma_{\mathbf{B}}$
      set $\Gamma = \Gamma'$
  else stop with "satisfiable"

In the LIA case, for example, the limit background contexts can always be made rigid-variable free by quantifier elimination. The set $\Gamma' \supset \Gamma$ can then be computed as $\Gamma' = \Gamma \cup \{\bigvee_{c \in \Gamma_{\mathbf{B}}} \neg c\}$, which is the weakest possible such set.

The derivation D in line 1 might not be finite, and hence line 2 might not be reached. In this case the procedure does not terminate, but this is acceptable as by the completeness theorem (Theorem 6.6) $\Psi \cup \Gamma$ is satisfiable then. Another source for non-termination comes from growing the sets Γ' without bound. This is theoretically acceptable, as our logic is not even semi-decidable. In practice, one could add to Γ finite domain declarations for all parameters involved, like $1 \leq a \leq 100$. This will obviously lead to finitely many Γ' only. Moreover, with the method suggested above, the sets Γ' will be computed in a conflict-driven way. For example, if $\Psi = \{P(x) \leftarrow x = a,\ \neg P(x) \leftarrow 1 \leq x \leq 100\}$ and $\Gamma = \{1 \leq a \leq 100\}$ then the procedure above will terminate with "unsatisfiable" in the first round, and the refutation will be found in $O(1)$ time. By contrast, finite-domain finders will essentially, work with the disjunction $a = 1 \vee \cdots \vee a = 100$. If $\Psi = \{P(x) \leftarrow x = a,\ \neg P(x) \leftarrow 1 \leq x \leq 50\}$ instead, a derivation (non-refutation) will be found in the second round that confines a to $51 \leq a \leq 100$.

Another special case is when all input clauses are of the form $C \leftarrow c \wedge x_1 = t_1 \wedge \cdots \wedge x_n = t_n$, where c and the t_i's are ground background terms. Because each t_i is ground, it cannot be instantiated in Split inferences, and by injectivity of assignments, Split can instantiate each x_i only with one single rigid variable, as no (satisfiable) background context can contain $v_1 = t$ and $v_2 = t$ for different v_1 and v_2. In consequence, the limit background context will always be finite, *for any input background context*, in particular those without any finite domain declarations for the parameters. Moreover, because the set of background terms is fixed a priori, there are only finitely many non-equivalent background contexts. Therefore, the procedure above cannot grow Γ' without bound, and the only source for nontermination is in line 1. One obvius way that guarantees termination for that as well is when the free variables in C are exactly $x_1, \ldots, x_n$. This will lead to ground foreground contexts only, which cannot grow infinitey due to calculus restrictions.

Notice that the clauses of this form are the image of abstraction of formulas of the

form $C \leftarrow c$ where C is a ground clause and c is a ground constraint. One may even add functionality axioms like e.g., $\neg P_f(x, y_1) \vee \neg P_f(x, y_2) \leftarrow y_1 \not\simeq y_2$, where P_f encodes a (unary) function symbol f. Adding these axioms does not endanger termination, because they can only be used to close branches, never to introduce fresh rigid variables into a foreground contexts. As a consequence, $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ can be used as a decision procedure for ground problems in the combination of the background theory and uninterpreted function symbols with equality (even B-sorted ones).

8 Conclusions

We presented the new $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ calculus, which properly generalizes the essentials of two earlier Model Evolution calculi, the $\mathcal{ME}_{\mathrm{E}}$ calculus with equality inference rules but without theory reasoning [5], and $\mathcal{ME}(\mathrm{LIA})$ [4], which is the other way round. We expect $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$-based theorem provers to be useful for problems that rely heavily on parameters. In Section 7 we provided some considerations why this is the case.

As always, much remains to be done. Among that is extension by "universal variables" and further simplification rules. As said before, we plan to extend the calculus directly with B-sorted (non-constant) function symbols. This could be done along the lines in [3]. Another pressing issue is to strengthen $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$'s model-building capabilities. For example, an input clause like $P(x) \leftarrow x > 0$ leads to nontermination, as, in essence, infinitely many instances $P(v_i) \leftarrow v_i > 0$ over rigid variables $v_1, v_2, \ldots$ are needed to represent the model of $P(x) \leftarrow x > 0$. This is clearly an undesirable situation. But other theorem proving calculi designed for the same logic face the same problem, if not on this example, only slightly more complicated ones will do. This indicates a serious research problem, which is beyond the scope of this paper.

References

[1] E. Althaus, E. Kruglov, and C. Weidenbach. Superposition modulo linear arithmetic sup(la). In S. Ghilardi and R. Sebastiani, editors, *FroCos*, volume 5749 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2009.

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

[3] L. Bachmair, H. Ganzinger, and U. Waldmann. Refutational theorem proving for hierachic first-order theories. *Appl. Algebra Eng. Commun. Comput*, 5:193–212, 1994.

[4] P. Baumgartner, A. Fuchs, and C. Tinelli. ME(LIA) – Model Evolution With Linear Integer Arithmetic Constraints. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming, Artificial*

Intelligence and Reasoning (LPAR'08), volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 258–273. Springer, November 2008.

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

[6] P. Baumgartner and U. Waldmann. Superposition and model evolution combined. In R. Schmidt, editor, *CADE-22 – The 22nd International Conference on Automated Deduction*, volume 5663 of *Lecture Notes in Artificial Intelligence*, pages 17–34, Montreal, Canada, July 2009. Springer.

[7] H. Bürckert. A Resolution Principle for Clauses with Constraints. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, LNAI 449, pages 178–192, Kaiserslautern, FRG, July 24–27, 1990. Springer-Verlag.

[8] H. Ganzinger and K. Korovin. Theory Instantiation. In *Proceedings of the 13 Conference on Logic for Programming Artificial Intelligence Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Computer Science*, pages 497–511. Springer, 2006.

[9] Y. Ge, C. Barrett, and C. Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In F. Pfenning, editor, *Proceedings of the 21st International Conference on Automated Deduction (CADE-21), Bremen, Germany*, Lecture Notes in Computer Science. Springer, 2007.

[10] K. Korovin and A. Voronkov. Integrating linear arithmetic into superposition calculus. In *Computer Science Logic (CSL'07)*, volume 4646 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2007.

[11] R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, Nov. 2006.

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

[13] P. Rümmer. A constraint sequent calculus for first-order logic with linear integer arithmetic. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'08)*, volume 5330 of *Lecture Notes in Artificial Intelligence*, pages 274–289. Springer, November 2008.

[14] M. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

A Proofs

The following lemma establishes an important relation between ground literals produced by Λ and the rewrite system $R_{\Lambda,\alpha}$.

Lemma A.1 *Let l and r be variable-free foreground terms and α a suitable assignment. If $l\alpha \succ r\alpha$. Then,*

(i) if $(l \to r)\alpha \in R_{\Lambda,\alpha}$ then Λ produces $l \approx r$, and

(ii) if $(l \to r)\alpha \notin R_{\Lambda,\alpha}$ and $l\alpha$ and $r\alpha$ are irreducible wrt $R_{\Lambda,\alpha}$ then Λ produces $\neg(l \approx r)$ and Λ does not produce $(l \approx r)$.

Proof. By stability of $\succ$ under assignments, $l \succ r$ entails $l\alpha \succ r\alpha$.

The statement (i) follows immediately from the definition of $R_{\Lambda,\alpha}$ and the fact that Λ produces a literal K if and only if $\Lambda\alpha$ produces $K\alpha$ (recall that α is injective).

Concerning (ii), suppose that $l\alpha$ and $r\alpha$ are irreducible wrt. $R_{\Lambda,\alpha}$. If Λ produces $l \approx r$ then $\Lambda\alpha$ produces $(l \approx r)\alpha$. We distinguish two cases. If $R_{\Lambda,\alpha}$ generates $(l \to r)\alpha$ then $l\alpha$ is reducible by $(l \to r)\alpha \in R_{\Lambda,\alpha}$. If $R_{\Lambda,\alpha}$ does not generate $(l \to r)\alpha$ then, by definition of $R_{\Lambda,\alpha}$, $l\alpha$ or $r\alpha$ must be reducible wrt. $(R_{\Lambda,\alpha})_{(l\approx r)\alpha}$, hence reducible wrt. $R_{\Lambda,\alpha}$. Both cases thus contradict the assumption that $l\alpha$ and $r\alpha$ are irreducible wrt. $R_{\Lambda,\alpha}$. It follows that Λ does not produce $l \to r$.

Thanks to the presence of the pseudo-literal $\neg x$ in every context, it is not difficult to see that every context produces K or $\overline{K}$, for every literal K. Thus, with Λ not producing $l \approx r$ we can conclude that Λ produces $\neg(l \approx r)$. □

By combining Definition 5.2 and Lemma A.1, we immediately conclude that Λ produces every literal in a variable-free context restriction R if $R_{\Lambda,\alpha} \models R\alpha$. The relevance of this result is that, whenever $R\alpha$ is satisfied by $R_{\Lambda,\alpha}$ then the "syntactic" notion of productivity can be used to identify such situations, independent of α, in particular to identify *relevant* Herbrand closures of clauses $C \leftarrow R \cdot c$ that are falsified by an induced interpretation $I[\Lambda, \pi, \alpha])$.

The completeness proof works consistently with relevant constrained clauses. The following result is instrumental in reducing a hypothetical relevant counterexample, one that is falsified in the induced model, to a smaller *relevant* counterexample that additionally preserves satisfaction of its context restriction. (For notions around redundancy, relevancy and preservation of satisfaction of context constraints is preserved by property (i) in Definition 5.3.)

Lemma A.2 (Inferences preserve relevant closures) *Let $\Lambda \cdot \Gamma \vdash \Phi$ be a sequent and α a suitable assignment. Assume given a* Pos-Res, Para *or* Ref *inference with selected clause $C \leftarrow R \cdot c$, selected context equation $l \approx r$ in case of* Para, *derived clause $C' \leftarrow R' \cdot c'$, and a ground instance via γ of this inference such that*

(i) $(C \leftarrow R \cdot c, \gamma)$ is a relevant closure wrt. Λ and α, and $(\Lambda, \alpha) \models (R, \gamma)$,

(ii-a) *in case of* Para*, where* σ *is the mgu of the given inference,* $l \approx r$ *produces* $(l \approx r)\alpha\sigma$ *in* $\Lambda\alpha$*,* $l \approx r$ *produces* $(l \approx r)\alpha\gamma$ *in* $\Lambda\alpha$*, and* $(l \approx r)\alpha\gamma$ *generates the rule* $(l \to r)\alpha\gamma$ *in* $R_{\Lambda,\alpha}$*, and*

(ii-b) *in case of* Pos-Res*, where* σ *is the context unifier of the given inference and* $K_i \in \Lambda$ *is the context literal paired with the clause literal* $L_i \in C$*,* K_i *produces* $\overline{L_i}\alpha\sigma$ *in* $\Lambda\alpha$*,* K_i *produces* $\overline{L_i}\alpha\gamma$ *in* $\Lambda\alpha$*, and* $\overline{L_i}\alpha\gamma$ *is irreducible wrt.* $R_{\Lambda,\alpha}$*.*

Then, $(C' \leftarrow R' \cdot c', \gamma)$ *is a relevant closure wrt.* Λ *and* α*, and* $(\Lambda, \alpha) \models (R', \gamma)$

Proof. For convenience we abbreviate $R := R_{\Lambda,\alpha}$ below.

With (i), by relevancy we have $R \models R\alpha\gamma$, i.e., with Definition 5.2, if $l \approx r \in R\alpha\gamma$ then $l \to r \in R$, and if $\neg(l \approx r) \in R\alpha\gamma$ then l and r are irreducible wrt. R. Moreover, $(\Lambda, \alpha) \models (R, \gamma)$ means that $R\alpha\gamma$ is non-trivial, and for every $l \approx r \in R\alpha$, if $l\gamma \succ r\gamma$ then l is not a variable, and for every $K \in R\alpha$, Λ produces K and $K\gamma$ by the same literal.

We have to show

(1) $R'\alpha\gamma$ is non-trivial, and for every $l \approx r \in R'\alpha$, if $l\gamma \succ r\gamma$ then l is not a variable,

(2) for every $K' \in R'\alpha$, Λ produces K' and $K'\gamma$ by the same literal,

(3) if $l \to r \in R'\alpha\gamma$ then $l \to r \in R$, and

(4) if $\neg(l \approx r) \in R'\alpha\gamma$ then l and r are irreducible wrt. R.

The property (1) is easily obtained from inspection of the inference rules. For the second part it is crucial that paramodulation into variables is forbidden. It remains to show (2), (3) and (4).

Let σ be the unifier as mentioned in case (ii-a) and (ii-b). Assume σ is idempotent, which is the case with usual unification algorithms. Because γ gives a ground instance of the given inference, γ must be a unifier for the same terms as σ. Because σ is a most general unifier, there is a substitution δ such that $\gamma = \sigma\delta$. With the idempotency of σ we get $\gamma = \sigma\delta = \sigma\sigma\delta = \sigma\gamma$.

For later use we prove some simple *facts*:

(i) if $K' \in R\alpha\sigma$ then Λ produces K' and $K'\gamma$ by the same literals.

Proof: Assume $K' \in R\alpha\sigma$ and let $K \in R\alpha$ such that $K\sigma = K'$. We already know that some $L \in \Lambda\alpha$ produces K in $\Lambda\alpha$ and L produces $K\gamma$ in $\Lambda\alpha$. If L didn't produce $K\sigma$ in $\Lambda\alpha$ then there would be a $L' \in \Lambda\alpha$ with $L \gtrsim \overline{L'} \gtrsim K\sigma$. With $\gamma = \sigma\delta$ and by transitivity of $\gtrsim$ we would get $L \gtrsim \overline{L'} \gtrsim K\gamma$, and so L would not produce $K\gamma$ either. Hence L produces $K\sigma$ in $\Lambda\alpha$. Because L produces $K\gamma$ in $\Lambda\alpha$, with $K\sigma = K'$ and $\gamma = \sigma\gamma$ conclude that L produces $K'\gamma$ in $\Lambda\alpha$, too.

(ii) if $l \approx r \in R\alpha\sigma\gamma$ then $l \to r \in R$.

Proof: we already know that if $l \to r \in R\alpha\gamma$ then $l \to r \in R$. The claim then follows immediately with $\gamma = \sigma\gamma$.

(iii) if $\neg(l \approx r) \in R\alpha\sigma\gamma$ then l and r are irreducible wrt. R.

Proof: we already know that if $\neg(l \approx r) \in R\alpha\gamma$ then l and r are irreducible wrt. R. The claim then follows immediately with $\gamma = \sigma\gamma$.

To prove (2), (3) and (4) we carry out a case analysis with respect to the inference rule applied.

In case of a Ref inference let the selected clause be $s \not\approx t \vee C'' \leftarrow R \cdot c$ and the derived clause $C' \leftarrow R' \cdot c' = (C'' \leftarrow R \cdot c)\sigma$. With $R' = R\sigma$, (2) follows directly from fact (i), (3) follows immediately from fact (ii), and (4) follows immediately from fact (iii).

In case of a Para inference let the selected clause be $C \leftarrow R \cdot c = [\neg](s[u]_p \approx t) \vee C'' \leftarrow R \cdot c$ and the conclusion $C' \leftarrow R' \cdot c' = ([\neg](s[r]_p \approx t) \vee C'' \leftarrow R \cup \{l \approx r\} \cdot c)\sigma$. The proofs of (2), (3) and (4) for the subset $R\sigma$ of R' follows immediately from facts (i), (ii) and (iii), respectively. Now consider the sole additional element $(l \approx r)\sigma$ that is in Γ' but not in $R\sigma$. Recall we are given that $l \approx r$ produces $(l \approx r)\alpha\sigma$ in $\Lambda\alpha$ and that $l \approx r$ produces $(l \approx r)_\alpha\gamma = (l \approx r)_\alpha\sigma\gamma$ in $\Lambda\alpha$, which proves (2). Regarding (3), recall we are given that $(l \approx r)\alpha\gamma$ generates $(l \to r)\alpha\gamma$ in R, which entails $(l \to r)\alpha\gamma = (l \to r)\alpha\sigma\gamma \in R$.

The proof for the case of Pos-Res is similar and is omitted. □

The ordering $\succ$ has already been extended to closures. For the purposes of the completeness proof, we work with evaluated closures, i.e., closures with some fixed assignment α applied to them. To this end, we introduce for any assignment α the parametrized ordering $\succ_\alpha$ as $s \succ_\alpha t$ iff $s\alpha \succ t\alpha$. Notice that $\succ_\alpha$ is total and well-founded on any set of variable-free foreground terms that α is suitable for, because α maps such terms to Herbrand terms, and $\succ$ is well-founded and total on Herbrand terms.

Now define $(C_1 \leftarrow R_1 \cdot c_1, \gamma_1) \succ_\alpha (C_2 \leftarrow R_2 \cdot c_2, \gamma_2)$ iff $((C_1 \leftarrow R_1 \cdot c_1)\alpha, \gamma_1) \succ ((C_2 \leftarrow R_2 \cdot c_2)\alpha, \gamma_2)$.

Because $\succ$ is stable under assignments (by definition) it follows $\mathscr{C} \succ_\alpha \mathscr{D}$ if $\mathscr{C} \succ \mathscr{D}$, for any Herbrand closures $\mathscr{C}$ and $\mathscr{D}$. Furthermore, with the remarks above, $\succ_\alpha$ is total and well-founded for any set of Herbrand closures that α is suitable for.

Definition A.3 (Smaller Relevant Closures from Φ wrt. Λ and α) Let Φ be a set of clauses, Λ a context, α an assignment, and $\mathscr{D}$ a Herbrand closure. Define

$$\begin{aligned}\Phi^{\Lambda,\alpha} = \{(C \leftarrow R \cdot c, \gamma) \mid\ & C \leftarrow R \cdot c \in \Phi \text{ and} \\ & (C \leftarrow R \cdot c, \gamma) \text{ is a relevant closure wrt. } \Lambda \text{ and } \alpha\}, \\ & \text{and} \\ \Phi^{\Lambda,\alpha}_{\mathscr{D}} = \{\mathscr{C} \in \Phi^{\Lambda,\alpha} \mid\ & \mathscr{D} \succ_\alpha \mathscr{C}\}\ .\end{aligned}$$

□

In words, $\Phi^{\Lambda,\alpha}_{\mathscr{D}}$ is the set of relevant closures wrt. Λ and α of all clauses from Φ that, when evaluated under α, are all smaller wrt. $\succ$ than $\mathscr{D}$ evaluated under α.

Lemma A.4 *Suppose* $(\pi, \alpha) \models \Gamma$. *If (i)* $(C \leftarrow R \cdot c, \gamma)$ *is redundant wrt.* $\Lambda \cdot \Gamma \vdash \Phi$ *and* $\mathscr{D}$*, (ii)* $(C \leftarrow R \cdot c, \gamma)$ *is a relevant closure wrt.* Λ *and* α*, and (iii)* $I[\Lambda, \pi, \alpha] \models \Phi^{\Lambda,\alpha}_{\mathscr{D}}$ *then* $I[\Lambda, \pi, \alpha] \models (C \leftarrow R \cdot c, \gamma)$.

Proof. Assume (i), (ii) and (iii). We have to show $I[\Lambda,\pi,\alpha] \models (C \leftarrow R\cdot c,\gamma)$.

If $(\Lambda,\alpha) \not\models (R,\gamma)$ then the conclusion follows trivially. Hence assume $(\Lambda,\alpha) \models (R,\gamma)$ from now on. From (ii) conclude $R_{\Lambda,\alpha} \models R\alpha\gamma$ by definition of relevance.

If case (a) or (b) in Definition 5.3 applies the claim holds trivially. Otherwise case (c) in Definition 5.3 gives us Herbrand closures $(C_i\cdot\Gamma_i,\gamma_i)$ of clauses $C_i\cdot\Gamma_i \in \Phi$ that satisfy conditions (i) – (iv) in Definition 5.3.

With $(\Lambda,\alpha) \models (R,\gamma)$ from property (i) in Definition 5.3 it follows $(\Lambda,\alpha) \models (R_i,\gamma_i)$. Likewise, with $R_{\Lambda,\alpha} \models R\alpha\gamma$ from property (i) in Definition 5.3 it follows $R_{\Lambda,\alpha} \models R_i\alpha\gamma_i$. Thus, each $(C_i \leftarrow R_i\cdot c_i,\ \gamma_i)$ is a relevant closure wrt. Λ and α. By condition (iii) in Definition 5.3, $(C_i \leftarrow R_i\cdot c_i,\ \gamma_i)$ is smaller wrt. $\succ$ than $\mathscr{D}$, and also smaller wrt. $\succ_\alpha$ than $\mathscr{D}$ by stability of $\succ$ under assignments. More formally, thus, $(C_i \leftarrow R_i\cdot c_i,\ \gamma_i) \in \Phi^{\Lambda,\alpha}_{\mathscr{D}}$, and with (iii) conclude $I[\Lambda,\pi,\alpha] \models (C_i \leftarrow R_i\cdot c_i,\ \gamma_i)$. With $(\Lambda,\alpha) \models (R_i,\gamma_i)$ from above it follows $I[\Lambda,\pi,\alpha] \models (C_i \leftarrow c_i)\gamma_i$.

Because each c_i is closed, $c_i\gamma_i = c_i$. If $(\pi,\alpha) \not\models c_i$ for some such c_i then from $(\pi,\alpha) \models \Gamma$ and (ii) conclude $(\pi,\alpha) \not\models c$ and using $c\gamma = c$ it follows (trivially) $I[\Lambda,\pi,\alpha] \models (C \leftarrow c)\gamma$.

If $(\pi,\alpha) \models c_i$ for each c_i, then from$I[\Lambda,\pi,\alpha] \models (C_i \leftarrow c_i)\gamma_i$ and the definition of induced interpretation it follows $R^\star_{\Lambda,\alpha} \models C_i\gamma_i\alpha$. By property (iv) of redundancy, $\{C_1\gamma_1,\ldots,C_n\gamma_n\} \models C\gamma$. Because α is injective it can be seen as a renaming of the rigid variables in these clauses into background domain elements, i.e, constants from a disjoint domain. Therefore $\{C_1\gamma_1\alpha,\ldots,C_n\gamma_n\alpha\} \models C\gamma\alpha$, and so $R^\star_{\Lambda,\alpha} \models C\gamma\alpha$. Again by induced interpretation $I[\Lambda,\pi,\alpha] \models (C \leftarrow c)\gamma$.

In both cases $I[\Lambda,\pi,\alpha] \models (C \leftarrow R\cdot c,\gamma)$ follows trivially from $I[\Lambda,\pi,\alpha] \models (C \leftarrow c)\gamma$ and we are done. □

Proposition A.5 *Suppose* $(\pi,\alpha) \models \Gamma$. *Let* $\Lambda\cdot\Gamma \vdash \Phi$ *be a sequent and* $(C \leftarrow R\cdot c,\gamma)$ *a Herbrand closure. If (i)* $C \leftarrow R\cdot c$ *is redundant wrt.* $\Lambda\cdot\Gamma \vdash \Phi$, *(ii)* $(C \leftarrow R\cdot c,\gamma)$ *is a relevant closure wrt.* Λ *and* α, *and (iii)* $I[\Lambda,\pi,\alpha] \models \Phi^{\Lambda,\alpha}_{(C\leftarrow R\cdot c,\gamma)}$ *then* $I[\Lambda,\pi,\alpha] \models (C \leftarrow R\cdot c,\gamma)$.

Proposition A.5 establishes a relationship between redundant clauses and satisfaction by $I[\Lambda,\pi,\alpha]$. It is used in the completeness proof below, which is based on an inductive argument that allows to assume that all relevant closures wrt. Λ and α that are smaller wrt. $\succ_\alpha$ than a hypothetically falsified closure $(C \leftarrow R\cdot c,\gamma)$ are all satisfied by $I[\Lambda,\pi,\alpha]$. Proposition A.5 then allows to conclude that $(C \leftarrow R\cdot c,\gamma)$ is satisfied by $I[\Lambda,\pi,\alpha]$, and hence cannot be that hypothetically falsified clause.

Proof. Immediate from Lemma A.4 by setting $\mathscr{D} = (C \leftarrow R\cdot c,\gamma)$, and using Definition 5.3. □

Theorem 5.6 (Static completeness) *Let* $\Lambda\cdot\Gamma \vdash \Phi$ *be a saturated sequent,* π *a valuation and* α *a suitable assignment for* $\Lambda\cdot\Gamma \vdash \Phi$.

If $(\pi,\alpha) \models \Gamma$, $\mathrm{ran}(\alpha) = |\mathsf{B}|$ *and* $(\square \leftarrow \emptyset\cdot\top) \notin \Phi$ *then the induced interpretation* $I[\Lambda,\pi,\alpha])$ *satisfies all Herbrand closures of all clauses in* Φ *that are relevant wrt.* Λ *and* α. *Moreover,* $I[\Lambda,\pi,\alpha] \models C \leftarrow c$, *for every* $C \leftarrow c \in \Phi$.

Proof. Suppose $(\pi,\alpha) \models \Gamma$, $\mathrm{ran}(\alpha) = |\mathsf{B}|$ and $(\square \leftarrow \emptyset\cdot\top) \notin \Phi$. For the proof of the first statement we show the following property (P):

(P) $I[\Lambda, \pi, \alpha] \models (C \leftarrow R \cdot c, \gamma)$,

for every relevant closure $(C \leftarrow R \cdot c, \gamma)$ wrt. Λ and α of every clause $C \leftarrow R \cdot c \in \Phi$.

Once (P) is shown, the "moreover" statement follows easily from the (trivial) facts that for clauses with empty context restrictions every Herbrand instance is trivially relevant and that $(\Lambda_\alpha, I[\alpha]) \models (C \leftarrow \emptyset \cdot c, \gamma)$ holds if and only if $I[\alpha] \models (C \leftarrow c)\gamma$.

We prove (P) by contradiction. Every counterexample, that is, every closure $(C \leftarrow R \cdot c, \gamma)$ of a clause $C \leftarrow R \cdot c \in \Phi$ that is relevant wrt. Λ and α and that does not satisfy (P) must satisfy the following properties:

(i) $R_{\Lambda,\alpha} \models R\alpha\gamma$, by relevancy.

(ii) $(\Lambda, \alpha) \models (R, \gamma)$, and

(iii) $I[\Lambda, \pi, \alpha] \not\models (C \leftarrow c)\gamma$, from $(C \leftarrow R \cdot c, \gamma)$ not satisfying (P) by Definition 3.6.

(iv) $(\pi, \alpha) \models c$, and

(v) $R^\star_{\Lambda,\alpha} \not\models C\alpha\gamma$, from (iii) and by definition of induced interpretation.

Among all counterexamples, by well-foundedness of the ordering $\succ_\alpha$ on Herband closures, there is a minimal counterexample (minimal wrt. $\succ_\alpha$). From now on let $(C \leftarrow R \cdot c, \gamma)$ be such a minimal counterexample.

By minimality of $(C \leftarrow R \cdot c, \gamma)$, every relevant closure of a clause in Φ wrt. Λ and α that is smaller wrt. $\succ_\alpha$ than $(C \leftarrow R \cdot c, \gamma)$ satisfies (P). More formally, $I[\Lambda, \pi, \alpha] \models \Phi^{\Lambda,\alpha}_{(C \leftarrow R \cdot c, \gamma)}$.

We carry out an exhaustive case analysis on properties of $(C \leftarrow R \cdot c, \gamma)$.

(1) $(C \leftarrow R \cdot c, \gamma)$ *is redundant wrt.* $\Lambda \cdot \Gamma \vdash \Phi$.
If $(C \leftarrow R \cdot c, \gamma)$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$, then by Lemma A.4, setting $\mathscr{D} = (C \leftarrow R \cdot c, \gamma)$ there, (P) follows immediately, contradicting our assumption. Hence, $(C \leftarrow R \cdot c, \gamma)$ cannot be redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$.

(2) $\mathrm{var}(C)\alpha\gamma$ *is reducible wrt.* $R_{\Lambda,\alpha}$.
The $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ calculus does not paramodulate into or below variables. To explain the completeness of this restriction we need to know that $\mathrm{var}(C)\alpha\gamma$ is irreducible wrt. $R_{\Lambda,\alpha}$.

First we show that every term $t \in (\mathrm{var}(C) \cap \mathrm{var}(R))\alpha\gamma$ is irreducible wrt. $R_{\Lambda,\alpha}$. For, if t is reducible wrt. $R_{\Lambda,\alpha}$ and occurs in a negative literal $\neg(l \approx r) \in R_\alpha\gamma$ then we get a contradiction to (i), as that negative literal is reducible wrt. $R_{\Lambda,\alpha}$. If t occurs in a positive literal $l \approx r \in R\alpha\gamma$ we conclude as follows: from (i) it follows $l \neq r$ and hence, w.l.o.g, $l \succ r$. As $l \approx r \in R\alpha\gamma$ there is a $l' \approx r' \in R\alpha$ such that $(l' \approx r')\gamma = l \approx r$. By Definition 3.5-(i), l' is not a variable. Hence, t occurs as a subterm of r or as a *proper* subterm of l. But then $l \to r$ is reducible by a smaller rule from $R_{\Lambda,\alpha}$, and hence $l \to r$ cannot be generated in $R_{\Lambda,\alpha}$, again contradicting (i).

If $x\alpha\gamma$ is reducible for some $x \in \mathrm{var}(C) \setminus \mathrm{var}(R)$, then a term in the range of $\gamma\alpha$ can be replaced by a smaller yet congruent term wrt. $R^\star_{\Lambda,\alpha}$. Observe that this results in a smaller (wrt. $\succ_\alpha$) counterexample, thus contradicting the choice of $(C \leftarrow R \cdot c, \gamma)$.

In summary, thus, $\mathrm{var}(C)\alpha\gamma$ is irreducible wrt. $R_{\Lambda,\alpha}$, which we may assume from now on.

(3) $C = s \not\approx t \vee D$.
Suppose that none of the preceding cases holds and $C = s \not\approx t \vee D$.

(3.1) $s\alpha\gamma = t\alpha\gamma$.
If $s\alpha\gamma = t\alpha\gamma$ then $s\gamma = t\gamma$ (because α is injective) and so there is a Ref inference with selected clause $(s \not\approx t \vee D \leftarrow R \cdot c)\gamma$ and derived clause $(D \leftarrow R \cdot c)\gamma$, which is a ground instance of a Ref inference with selected clause $s \not\approx t \vee D \leftarrow R \cdot c$ and derived clause $(D \leftarrow R \cdot c)\sigma$. It is safe to assume that σ is idempotent, which gives us $\sigma\gamma = \gamma$.

By saturation, that Ref inference is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$. Because the closure $(C \leftarrow R \cdot c, \gamma)$ is not redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$, the derived clause, taken as the closure $\mathscr{C} := ((D \leftarrow R \cup \{l' \approx r'\} \cdot c)\sigma, \gamma)$. must be redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ and $(C \leftarrow R \cdot c, \gamma)$ by definition of redundant inferences. Furthermore, with Lemma A.2 it is a relevant closure wrt. Λ and α, hence, by Lemma A.4, $I[\Lambda, \pi, \alpha] \models \mathscr{C}$. By Definition 3.6, this means $(\Lambda, \alpha) \not\models (R \cup \{l' \to r'\})\sigma, \gamma)$ or $I[\Lambda, \pi, \alpha] \models (D \leftarrow c)\sigma\gamma$. However, Lemma A.2 gives us additionally $\Lambda \models ((R \cup \{l' \approx r'\})\sigma, \gamma)$, and so the former case is impossible. But then, from $I[\Lambda, \pi, \alpha] \models (D \leftarrow c)\sigma\gamma$ and $\sigma\gamma = \gamma$ by definition of induced interpretation it follows $(\pi, \alpha) \not\models c$ or $R^\star_{\Lambda,\alpha} \models D\alpha\gamma$. However, by (iv) the first case is impossible, and in the second case, trivially, $R^\star_{\Lambda,\alpha} \models C\alpha\gamma$, a plain contradiction to (v) above.

(3.2) $s\alpha\gamma \neq t\alpha\gamma$.
If $s\alpha\gamma \neq t\alpha\gamma$ then without loss of generality assume $s\alpha\gamma \succ t\alpha\gamma$. With (v) it follows $R^\star_{\Lambda,\alpha} \models (s \approx t)\alpha\gamma$. Because $R_{\Lambda,\alpha}$ is a convergent rewrite system, $s\alpha\gamma$ and $t\alpha\gamma$ must have the same normal forms. In particular, thus, $s\alpha\gamma$ is reducible wrt. $R_{\Lambda,\alpha}$. Suppose $s\alpha\gamma = (s\alpha\gamma)[l\alpha]_p$ for some position p and ground terms l and r such that $(l \to r)\alpha \in R_{\Lambda,\alpha}$. Because α is injective it follows $s\gamma = (s\gamma)[l]_p$

With Lemma A.1-(i) it follows that Λ produces $l \approx r$. For later use let $l' \approx r' \in \Lambda$ be a fresh variant that produces $l \approx r$ in Λ and assume that γ has already been extended so that $(l' \approx r')\gamma = l \approx r$.

The conclusions so far give that Para is applicable with selected context equation $(l' \approx r')\gamma$, selected clause $s\gamma[l'\gamma]_p \not\approx t\gamma \vee D\gamma \leftarrow R\gamma \cdot c$[7] and derived clause $s\gamma[r'\gamma]_p \not\approx t\gamma \vee D\gamma \leftarrow R\gamma \cup \{(l' \approx r')\gamma\} \cdot c$. The next step is to show that this inference is a ground instance via γ of a Para inference with selected context equation $l' \approx r'$, selected clause $C \leftarrow R \cdot c = s[u]_p \not\approx t \vee D \leftarrow R \cdot c$ and derived clause $(s[r']_p \not\approx t \vee D \leftarrow R \cup \{l' \approx r'\} \cdot c)\sigma$, where σ is an mgu of l' and u.

The position p in $s\gamma$ cannot be at or below a variable position in s, because otherwise we had $x\gamma[l'\gamma]_p$ for some variable x occuring in s, and so $x_\alpha\gamma$ would be reducible by $(l' \to r')\alpha\gamma = (l \to r)\alpha$, which is impossible by case (2) above. Hence, the position p exists in s, and the term u at that position is not a variable. Then it follows easily that the mgu σ of l' and u exists. Furthermore, the rules in $R_{\Lambda,\alpha}$ are all between F-sorted Herbrand terms, and it follows that l' (and r') are F-sorted, and hence u cannot be a rigid variable, as rigid variables and F-sorted terms are syntactically different, and hence do not unify. Finally, as

[7] As c is closed we have $c\gamma = c$.

all rules in $R_{\Lambda,a}$ are ordered wrt. $\succ$, from $(l \to r)\alpha \in R_{\Lambda,\alpha}$ it follows $r'\sigma \not\succeq l'\sigma$. Alltogether, we have established now that the claimed Para inference exists.

It is safe to assume that σ is idempotent, which gives us $\sigma\gamma = \gamma$.

By saturation, the Para inference is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$. Because the closure $(C \leftarrow R \cdot c, \gamma)$ is not redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$, the derived clause, taken as the closure $\mathscr{C} := ((s[r']_p \not\approx t \vee D \leftarrow R \cup \{l' \approx r'\} \cdot c)\sigma, \gamma)$. must be redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ and $(C \leftarrow R \cdot c, \gamma)$ by definition of redundant inferences. Furthermore, with Lemma A.2 it is a relevant closure wrt. Λ and α, hence, by Lemma A.4, $I[\Lambda, \pi, \alpha] \models \mathscr{C}$. By Definition 3.6, this means $(\Lambda, \alpha) \not\models (R \cup \{l' \to r'\})\sigma, \gamma)$ or $I[\Lambda, \pi, \alpha] \models (s[r']_p \not\approx t \vee D \leftarrow c)\sigma\gamma$. However, Lemma A.2 gives us additionally $\Lambda \models ((R \cup \{l' \approx r'\})\sigma, \gamma)$, and so the former case is impossible. But then, from $I[\Lambda, \pi, \alpha] \models (s[r']_p \not\approx t \vee D \leftarrow c)\sigma\gamma$ by definition of induced interpretation it follows $(\pi, \alpha) \not\models c$ (observe that $c\sigma\gamma = c$, as c is closed) or $R^\star_{\Lambda,\alpha} \models (s[r']_p \not\approx t \vee D)\alpha\sigma\gamma$. However, by (iv) the first case is impossible. In the second case, with $(l' \to r')\alpha\gamma \in R_{\Lambda,\alpha}$ by congruence and $\sigma\gamma = \gamma$ it follows $R^\star_{\Lambda,\alpha} \models (s \not\approx t \vee D)\alpha\gamma$. Using $C = s \not\approx t \vee D$ this is a plain contradiction to (v) above.

(4) $C = s \approx t \vee D$.
Suppose that none of the previous cases applies. This entails that C cannot contain a negative literal. In case (4) we examine the case that C is not empty, i.e., that C consists of positive literals only, at least one.

Suppose $C = s \approx t \vee D$. With (v), $s_\alpha\gamma = t_\alpha\gamma$ and hence also $s\gamma = t\gamma$ is impossible. Hence assume $s\gamma \neq t\gamma$ in the following. We distinguish two further cases.

(4.1) $s\alpha\gamma$ or $t\alpha\gamma$ is reducible wrt. $R_{\Lambda,\alpha}$.
If $s\alpha\gamma$ or $t\alpha\gamma$ is reducible wrt. $R_{\Lambda,\alpha}$ then there are ground terms l and r such that rule $(l \to r)\alpha \in R_{\Lambda,\alpha}$ that rewrites $s\alpha\gamma$ or $t\alpha\gamma$. With the same argumentation as in case (3.2) one shows that a smaller, yet congruent counterexample would exists due to a Para inference, which leads to the same contradiction as in case (3.2).

(4.2) $s\alpha\gamma$ and $t\alpha\gamma$ are irreducible wrt. $R_{\Lambda,\alpha}$.
This case is meant to say that case (4.1) does not apply, for no equation $s \approx t \in C$. Thus, for every $s \approx t \in C$, $s\alpha\gamma$ and $t\alpha\gamma$ are irreducible wrt. $R_{\Lambda,\alpha}$. Assume, w.l.o.g., $s\alpha\gamma \succ t\alpha\gamma$. With Lemma A.1-(ii) then conclude that some literal $\neg A \in \Lambda$ produces $\neg(s \approx t)\gamma$ in Λ. This indicates that a Pos-Res inference exists. More precisely, the selected clause of that inference is $(C \leftarrow R \cdot c)\gamma$ and the derived clause is $(\Box \leftarrow (R \cup \overline{C}) \cdot c)\gamma$. It is routine by now to check that this Pos-Res inference is a ground instance via γ of a Pos-Res inference with selected clause $C \leftarrow R \cdot c$ and derived clause $(\Box \leftarrow (R \cup \overline{C}) \cdot c)\sigma$, where σ is the most general simultaneous unifier of that inference.

The rest of the proof uses the same arguments as in case (3.2) and is omitted (we just note that $(C \leftarrow R \cdot c)\gamma \succ_\alpha (\Box \leftarrow (R \cup \overline{C}) \cdot c)\gamma$ and hence the latter clause is satisfied in $I[\Lambda, \pi, \alpha]$ by induction, which will again lead to a contradiction to (v)).

(5) $C = \Box$.
Suppose $C = \Box$. First we are going to show that Split is applicable to $\Lambda \cdot \Gamma \vdash \Phi$ with selected clause $\Box \leftarrow R \cdot c$.

With property (ii) above, $\Lambda\alpha$ produces every literal in $R\alpha$ and every literal in $R\alpha\gamma$. More precisely, for every $K' \in R\alpha$ there is a $L \in \Lambda\alpha$ that produces K in $\Lambda\alpha$ (and that produces $K\gamma$ in $\Lambda\alpha$). As said earlier, from properties of assignments it follows that L produces K in Λ (and that produces $K\gamma$ in Λ) , where $K \in R$ such that $K\alpha = K'$ and $L \in \Lambda$ such that $L\alpha = L'$ (*).

If Close were applicable with selected clause $\square \leftarrow R \cdot c$, then, by saturation, this Close inference is redundant, which is the case only if $\square \leftarrow \emptyset \cdot \top \in \Phi$. However, we are given that $\square \leftarrow \emptyset \cdot \top \notin \Phi$, and hence Close is not applicable. Together with $(\pi, \alpha) \models \Gamma$, as given, and (iv) it follows that $\Gamma \cup \{c\}$ is satisfiable, and therefore $R \not\subseteq \Lambda$. Thus, there is a literal $K \in R$ such that $K \notin R$. In other words, $\overline{K}$ is not contradictory with Λ.

In order to show that Split is applicable we need to show, additionally, that K is not contradictory with Λ. For, if it were, this means that $\overline{K} \in \Lambda$, and then Λ produces $\overline{K}$. But, as all derived contexts are not contradictory, Λ cannot produce K then, plainly contradicting (*) above. Notice that the background context $\Gamma^\star$ required for the right conclusion of Split exists trivially, as a consequence of condition (4), albeit not in a constructive way.

Thus, at this stage we know that Split is applicable with selected clause $\square \leftarrow R \cdot c$ and split literal $\overline{K}$.

By redundancy then, (a) there is a literal $L \in R$ such that Λ does not produce L in Λ, or (b) $\overline{K}$ is contradictory with Λ. The case (a) plainly contradicts (*), and case (b) plainly contradicts an earlier conclusion that $\overline{K}$ is not contradictory with Λ. □

Lemma A.7 *If $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$, Γ' is obtained from Γ by adding closed constraints, Λ' is obtained from Λ by adding non-contradictory rewrite literals, and Φ' is obtained from Φ by deleting clauses that are redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ and/or by adding arbitrary clauses, then $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda' \cdot \Gamma' \vdash \Phi'$.*

Proof. It is obvious from Def. 5.3 that a clause that is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ remains redundant if a closed constraint is added to Γ (by monotonicity of first-order logic) or an arbitrary clause is added to Φ, if a non-contradictory literal is added to Λ one needs to prove that, in terms of Def. 5.3, $(C \leftarrow R \cdot c, \gamma)$ remains redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ and $\mathscr{D}$. This is straightforward to check.

To prove that a clause that is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ remains redundant if redundant clauses are deleted from Φ, it suffices to show that the clauses $C_i \leftarrow R_i \cdot c_i \in \Phi$ in Definition 5.3 can always be chosen in such a way that they are not themselves redundant or their deletion does not affect redundancy of $C \leftarrow R \cdot c$: Suppose that a closure $(C \leftarrow R \cdot c, \gamma)$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ and $\mathscr{D}$. Let $\{(C_i \leftarrow R_i \cdot c_i, \gamma_i) \mid 1 \leq i \leq n\}$ be a minimal set of closures of clauses in Φ (wrt. the multiset extension of the clause ordering) that satisfies the conditions of Definition 5.3. Suppose that one of the $(C_i \leftarrow R_i \cdot c_i, \gamma_i)$, say $(C_1 \leftarrow R_1 \cdot c_1, \gamma_1)$, is redundant itself.

If case (a) in Definition 5.3 applies to $(C_1 \leftarrow R_1 \cdot c_1, \gamma_1)$ then from Definition 5.3-(i) it follows that $(C \leftarrow R \cdot c, \gamma)$ is redundant, too. Otherwise there exist Herbrand closures $(C_{1i} \leftarrow R_{1i} \cdot c_{1i}, \gamma_{1i})$ of clauses $C_{1i} \leftarrow R_{1i} \cdot c_{1i} \in \Phi$ that satisfy the conditions of Definition 5.3 for $(C_1 \leftarrow R_1 \cdot c_1, \gamma_1)$. But then $\{(C_i \leftarrow R_i \cdot c_i, \gamma_i) \mid 2 \leq i \leq n\} \cup \{(C_{1i} \leftarrow R_{1i} \cdot c_{1i}, \gamma_{1i}) \mid 1 \leq$

$i \leq m\}$ would also satisfy the conditions of Definition 5.3 for $(C \leftarrow R \cdot c, \gamma)$, contradicting the minimality of $\{(C_i \leftarrow R_i \cdot c_i, \gamma_i) \mid 1 \leq i \leq n\}$. □

Lemma A.8 *If a* Pos-Res, Ref *or* Para *inference is redundant wrt.* $\Lambda \cdot \Gamma \vdash \Phi$, Γ' *is obtained from* Γ *by adding closed constraints,* Λ' *is obtained from* Λ *by adding non-contradictory literals, and* Φ' *is obtained from* Φ *by deleting clauses that are redundant wrt.* $\Lambda \cdot \Gamma \vdash \Phi$ *and/or by adding arbitrary clauses, then this* Deduce *inference is redundant wrt.* $\Lambda' \cdot \Gamma' \vdash \Phi'$.

Proof. The non-trivial case is, in terms of Definition 5.4, to show that $(C' \leftarrow R' \cdot c', \gamma)$ remains redundant under the stated modifications of $\Lambda \cdot \Gamma \vdash \Phi$. This is shown analogously to the proof of Lemma A.7. □

Lemma A.9 *Let* $C \leftarrow R \cdot c$ *be a clause. If* $C \leftarrow R \cdot c$ *is redundant wrt.* $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$, *for some* $j < \kappa$, *then* $C \leftarrow R \cdot c$ *is redundant wrt.* $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$.

Proof. As a convenience, we denote the union of all clauses of a branch $\mathbf{B}$ by $\Phi^+_{\mathbf{B}} = \bigcup_{i<\kappa} \Phi_i$.

Suppose that $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$. Since $\Gamma_{\mathbf{B}} \supseteq \Gamma_j$, $\Lambda_{\mathbf{B}} \supseteq \Lambda_j$ and $\Phi^+_{\mathbf{B}} \supseteq \Phi_j$, Lemma A.7 implies that $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi^+_{\mathbf{B}}$. Now observe that every clause in $\Phi^+_{\mathbf{B}} \setminus \Phi_{\mathbf{B}}$ has been deleted at some node of the branch $\mathbf{B}$, which is only possible if it was redundant wrt. some $\Lambda_k \cdot \Gamma_k \vdash \Phi_k$ with $k < \kappa$. Again using Lemma A.7, we see that every clause in $\Phi^+_{\mathbf{B}} \setminus \Phi_{\mathbf{B}}$ is redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi^+_{\mathbf{B}}$. Hence $\Phi_{\mathbf{B}}$ is obtained from $\Phi^+_{\mathbf{B}}$ by deleting redundant clauses. By using Lemma A.7 a third time, we conclude that $C \leftarrow R \cdot c$ is redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. □

Lemma A.10 *Every* Pos-Res, Ref *or* Para *inference that is redundant wrt.* $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$, *for some* $j < \kappa$, *is redundant wrt.* $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$.

Proof. Analogously to the proof of Lemma A.9 using Lemma A.8. □

Proposition 6.5 (Exhausted Branches are Saturated) *If* $\mathbf{B}$ *is an exhausted branch of a limit tree of a fair derivation then* $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ *is saturated.*

Proof. Suppose $\mathbf{B}$ is an exhausted branch of a limit tree of some fair derivation. We have to show that every $\mathcal{ME}_{\mathrm{E}}(\mathrm{T})$ inference with a core inference rule and premise $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. We do this by assuming such an inference and carrying out a case analysis wrt. the inference rule applied.

By Definition 6.1 there is no Close inference with premise $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$, for no $i < \kappa$, with a persistent selected clause. But then there is no Close inference with premise $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ either. (Because if so, for a large enough i there would be Close inference with premise $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$, which we excluded.) Thus there is nothing to show for Close.

If the inference rule is Split then let $\square \leftarrow R \cdot c$ be the selected clause. There are only finitely many literals L, modulo renaming and modulo sign, that are more general wrt. the instantiation preorder $\gtrsim$ than a given literal K. The applicability conditions of the Split inference rule, the only rule that can add literals to foreground contexts, makes sure that

from some time k onwards, no more such literal L will be added to $\Lambda_k, \Lambda_{k+1}, \ldots$. Suppose k is chosen large enough so that this is the case for all literals $K \in R$.

We are given that $\square \leftarrow R \cdot c$ is persistent. Therefore suppose also $\square \leftarrow R \cdot c \in \Phi_k, \Phi_{k+1}, \ldots$, or choose k big enough. Together this shows that every Split inference with premise $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ already exists with with premise $\Lambda_k \cdot \Gamma_k \vdash \Phi_k$ and same selected clause, and vice versa. Recall from the definition of Split that Λ_k must produce every literal in R.

By Definition 6.1, the Split inference is redundant wrt. $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$, for some $j < \kappa$ with $j \geq k$. By redundancy, the selected clause $\square \leftarrow R \cdot c$ is redundant wrt. $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$, and, with Lemma A.9, redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. Or, (a) there is a literal $K \in R$ such that Λ_j does not produce K or (b) the split literal is contradictory with Λ_j. The case (a) is impossible because from time k onwards no more general literal than K (modulo sign) is added to Λ_k, and so Λ_j cannot contain a literal that would prevent Λ_j from producing K as well. In case (b), the split literal is also contradictory with $\Lambda_{\mathbf{B}}$ (as $\Lambda_{\mathbf{B}} \supseteq \Lambda_j$), and hence the assumed Split inference would not exist.

If the inference rule is Pos-Res, Ref or Para then by Definition 6.1 the inference is redundant wrt. $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$, for some $j \geq i$, and by Lemma A.10 it is redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. □

Proposition 6.3 (Compactness of finitely committed branches) *If* $\mathbf{B}$ *is finitely committed then there is a* π *and an* α *such that* $\mathrm{ran}(\alpha) = |\mathsf{B}|$ *and* $(\pi, \alpha) \models \Gamma_{\mathbf{B}}$.

Proof. Let $\mathbf{B}$ as stated. If $\Gamma_{\mathbf{B}}$ is finite, let $\pi_{\mathbf{B}}$ and $\alpha_{\mathbf{B}}$ such that $(\pi_{\mathbf{B}}, \alpha_{\mathbf{B}}) \models \Gamma_{\mathbf{B}}$, which must exist as all derivable background contexts are satisfiable by definition of the inference rules. Take $\pi = \pi_{\mathbf{B}}$ and obtain α from $\alpha_{\mathbf{B}}$ by composition with any injective mapping from the rigid variables onto $|\mathsf{B}| \setminus \mathrm{ran}(\alpha_{\mathbf{B}})$. It is easy to see that π and α suffice to prove the claim.

Hence suppose from now on that $\Gamma_{\mathbf{B}}$ is infinite. Let $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ be the sequent labelling the node i in the branch $\mathbf{B}$, for all $i < \kappa$.

By Definition 6.2 there are valuations π_i and assignments α_i such that $(\pi_i, \alpha_i) \models \Gamma_i$, for all $i < \kappa$.

Take an arbitrary background domain element $n \in |\mathsf{B}|$. By Definition 6.2-(ii), the set $\mathrm{pre}(n) := \bigcup_{i<\kappa} \mathrm{pre}_{\alpha_i}(n)$ is finite (it is the set of rigid variables that represent n in the α_i's). Moreover, with Definition 6.2-(i) it follows there is an i such that for all $j \geq i$ there is a $v \in \mathrm{pre}(n)$ such that $\alpha_j(v) = n$ (*).

Consider an arbitrary rigid variable v that occurs in $\Gamma_{\mathbf{B}}$. By Definition 6.2-(iii), the set $\mathrm{img}(v) := \bigcup_{i<\kappa} \mathrm{img}_{\alpha_i}(v)$ is finite (it is the set of background domain elements that v is mapped to in the α_i's). Because v occurs in $\Gamma_{\mathbf{B}}$ and because background contexts grow monotonically, there is an i such that for all $j \geq i$, v occurs in Γ_i. Together, thus, there is an i such that for all $j \geq i$ it holds that $\alpha_j(v) \in \mathrm{img}(v)$ (**).

Consider an arbitrary parameter a that occurs in $\Gamma_{\mathbf{B}}$. By Definition 6.2-(iv), the set $\mathrm{img}(a) := \bigcup_{i<\kappa} \mathrm{img}_{\pi_i}(a)$ is finite. Because a occurs in $\Gamma_{\mathbf{B}}$ and because background contexts grow monotonically, there is an i such that for all $j \geq i$, a occurs in Γ_i. Together, thus, there is an i such that for all $j \geq i$ it holds that $\pi_j(a) \in \mathrm{img}(a)$ (***).

Observe that for (*), (**) and (***) the same start index i can be chosen.

The next step is to define the desired valuation π and assignment α. Let $K = \{1, 2, \ldots\}$ be an index set consisting initially of all natural numbers. We consider sequences $(\Gamma_i)_{i \in K}$ and their associated valuations π_i and assignments α_i. Below we describe how to "thin out" K by iteratively taking subsets of K, which will provide π after finitely many steps, and α in the limit.

The first step is to fix π this way. Let A be the set of parameters that occurs in $\Gamma_{\mathbf{B}}$ and initially $K = \{1, 2, \ldots\}$. For a given $a \in A$ there is an infinite subset $\{k_1, k_2, \ldots\} \subseteq K$, a *thinning* of K, such that $k_1, k_2, \ldots \geq i$ and such that $n = \pi_{k_1}(a) = \pi_{k_2}(a) = \cdots$, for some $n \in \mathrm{img}(a)$. Such a thinning and must exist by (***) (recall that $\mathrm{img}(a)$ is finite, but there are infinitely many indices $k \in K$ with $k \geq i$, and so among all these valuations α_k there must be infinitely many that map a to n.)

As input clause sets are finite and there are no inference rules that introduce parameters, the set A is finite. This allows us to iterate the procedure just describe for all parameters in A. The result will be an infinite subset K_A of K such that $\pi_{k_1}(a) = \pi_{k_2}(a) = \cdots$ for all $a \in A$ where $K_A = \{k_1, k_2, \ldots\}$. Let π be any valuation that extends π_{k_1} to the paramaters not in A.

The next step is to determine α, starting with K_A. This is done by simultaneously taking a domain element from $|\mathsf{B}|$ and a rigid variable that occurs in $\Gamma_{\mathbf{B}}$, one pair after the other, and chosing rigid variables to represent the elements from $|\mathsf{B}|$, and chosing background domain elements to map the rigid variables from $\Gamma_{\mathbf{B}}$ to. These choices are guided by the satisfying assignments (π_i, α_i) for Γ_i, where i is taken from the current index set K, initially K_A. After each such choice, K is "thinned out" by keeping only those (infinitely many) indices that enforce compatibility of all subsequent choices with the current choice.

Let $K = K_A$ now. For a given $n \in |\mathsf{B}|$ there is an infinite subset $\{k_1, k_2, \ldots\} \subseteq K$ such that $k_1, k_2, \ldots \geq i$ and such that $n = \alpha_{k_1}(v) = \alpha_{k_2}(v) = \cdots$, for some $v \in \mathrm{pre}(n)$ (****). Such a set $\{k_1, k_2, \ldots\}$ and a rigid variable v must exist by (*) (recall that $\mathrm{pre}(n)$ is finite, but there are infinitely many indices $k \in K$ with $k \geq i$, and so among these assignments α_k there must be infinitely many that map some element $v \in \mathrm{pre}(n)$ to n).

Let V be the set of rigid variables that occurs in $\Gamma_{\mathbf{B}}$. For a given $v \in V$ there is an infinite subset $\{k_1, k_2, \ldots\} \subseteq K$ such that $k_1, k_2, \ldots \geq i$ and such that $n = \alpha_{k_1}(v) = \alpha_{k_2}(v) = \cdots$, for some $n \in \mathrm{img}(v)$ (*****). The arguments are similar as above, this time using (**) instead of (*).

For a given v and n selected either way, we refer to the set $\{k_1, k_2, \ldots\}$ as a *thinning of* K *to* (v, n).

We are now ready to describe a procedure that yields, in the limit, the desired assignment α. It proceeds as follows.

Set $N := |\mathsf{B}|$
Set $K := \{1, 2, \ldots\}$
Set $D := \{v \mid v \text{ is a rigid variable that occurs in } \Gamma_{\mathbf{B}}\}$
loop:
    Chose some $n \in N$
    Let K_n be a thinning of K to (v', n), for some $v' \in \mathrm{pre}(n)$
    Set $N := N \setminus \{n\}$, $V := V \setminus \{v'\}$, and $K := K_n$
    Chose some $v \in V$

Let K_v be a thinning of K to (v, n'), for some $n' \in \mathrm{img}(v)$
Set $N := N \setminus \{n'\}$, $V := V \setminus \{v\}$, and $K := K_v$
goto loop

Observe that, given n, a thinning K_n of K to (v', n) for some $v' \in \mathrm{pre}(n)$ exists by the argumentation above whenever K is infinite, and similarly for a given v and thinning K_v of K to (v, n') Notice that initially K is infinite, and thus infiniteness of K is preserved as an invariant. This shows that the above procedure is well-defined.

The next step is to define the desired assignment α as a collection of all the pairs (v, n) mentioned in the loop body. To this end, we show first that there are no such pairs (v, n) and (v, n') with $n \neq n'$ (to guarantee functionality), and that there are no pairs (v, n) and (v', n) with $v \neq v'$ (to guarantee injectivity).

Suppose the procedure has obtained K' as a thinning of K to (v, n), for some v and n. Then, by construction n is removed from N and v is removed from V. Therefore, the procedure cannot chose n (in the first part) or v (in the second part) for thinning in the future.

Should the procedure consider a thinning to (v, n') in the future with $n \neq n'$, thus, this could only be because in the first part n' is chosen and the thinning is to (v, n'). But then, from (****) and (*****) and by consruction of thinnings there would be an α_i such that $\alpha_i(v) = n$ and $\alpha_i(v) = n'$, which is impossible by functionality of assignments.

Similarly, should the procedure consider a thinning to (v', n) in the future with $v \neq v'$, thus, this could only be because in the second part v' is chosen and the thinning is to (v', n). But then, again from (****) and (*****) and by construction of thinnings there would be an α_i such that $\alpha_i(v) = n$ and $\alpha_i(v') = n$, which is impossible by injectivity of assignments.

By construction, the procedure above defines a pair (v', n) for every $n \in N$ and a pair (v, n') for every rigid variable v occuring in $\Gamma_{\mathbf{B}}$. With the functionality and injectivity properties already shown above, we can define $\alpha(v) = n$ for every such pair, and it holds that α is well-defined (as a function) and injective, that its range includes N and that it is defined on every rigid variable occurring in $\Gamma_{\mathbf{B}}$, i.e., α is suitable for $\Gamma_{\mathbf{B}}$.

Finally, it remains to show $(\pi, \alpha) \models c$, for every $c \in \Gamma_{\mathbf{B}}$. We do this by contradiction.

Suppose $(\pi, \alpha) \not\models c$, for some $c \in \Gamma_{\mathbf{B}}$. By definition, $c \in \Gamma_{\mathbf{B}}$ means that $c \in \Gamma_j$, for some i and all $j \geq i$. The constraint c can contain only finitely many rigid variables, say $v_1, \ldots, v_m$. As α is suitable for $\Gamma_{\mathbf{B}}$, α is defined on all $v_1, \ldots, v_m$. Say, $\alpha(v_1) = n_1, \ldots, \alpha(v_m) = n_m$. By construction of α, the procedure must have derived the corresponding tuples $(v_1, n_1), \ldots, (v_n, n_m)$ (in some order). Once more with (****) and (*****) and by construction of thinnings there would be an α_i such that $\alpha_i(v_1) = n_1, \ldots, \alpha_i(v_m) = n_m$.

Chosing i large enough we may assume $c \in \Gamma_i$. By definition of π_i and α_i, $(\pi_i, \alpha_i) \models \Gamma_i$ and hence $\alpha_i \models c$. However, c is a closed constraint, and α and α_i are the same function on all rigid variables in c. Recall that the procedure to define α starts with the index set K_A and $\pi_j(a) = \pi(a)$ by construction, for all $j \in K_A$ and $a \in A$, and hence $\pi_i(a) = \pi(a)$. We thus get a contradiction to the assumption $(\pi, \alpha) \not\models c$. Hence, $(\pi, \alpha) \models c$, for every $c \in \Gamma_{\mathbf{B}}$. □

Theorem 6.6 (Completeness) *Let Ψ be a set of input clauses, Γ a satisfiable set of closed constraints and $\mathbf{T}$ be the limit tree of a fair derivation from Ψ and Γ that is not a refutation*

tree. Let $\mathbf{B}$ *be any exhausted and finitely committed branch of* $\mathbf{T}$*, and* $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$ *the limit sequent of* $\mathbf{B}$*.*

Let π *be a valuation and* α *a suitable assignment for* $\Gamma_\mathbf{B}$ *with* $\mathrm{ran}(\alpha) = |\mathsf{B}|$ *such that* $(\pi, \alpha) \models \Gamma_\mathbf{B}$*, which exist by Proposition 6.3. Then, the induced interpretation* $\mathcal{I}[\Lambda_\mathbf{B}, \pi, \alpha]$ *satisfies all Herbrand closures of all clauses in* $\Phi_\mathbf{B}$ *that are relevant wrt.* $\Lambda_\mathbf{B}$ *and* α*. Moreover,* $\mathcal{I}[\Lambda_\mathbf{B}, \pi, \alpha] \models \Psi$*.*

Proof. Suppose $\mathbf{T}$ is not a refutation tree and let $\mathbf{B}$ an exhausted branch of $\mathbf{T}$. By Proposition 6.5 the limit sequent $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$ is saturated. By Theorem 5.6 we get the first conclusion immediately.

For the "moreover" part, let $C \leftarrow c$ be any clause from Ψ. If $C \leftarrow c \in \Phi_\mathbf{B}$ then with the second part of Theorem 5.6 nothing remains to be shown. Otherwise $C \leftarrow c \notin \Phi_\mathbf{B}$. Hence $C \leftarrow c$ has been removed at some time $k < \kappa$ from the clause set Φ_k of the sequent $\Lambda_k \cdot \Gamma_k \vdash \Phi_k$ by an application of the Simp rule. By definition of Simp, $C \leftarrow c$ is redundant wrt. $\Lambda_{k+1} \cdot \Gamma_{k+1} \vdash \Phi_{k+1}$. By Lemma A.9, $C \leftarrow c$ is redundant wrt. $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$. Since $C \leftarrow c$ has an empty constraint, all its closures are relevant, and by Proposition A.5, all relevant closures wrt. $\Lambda_\mathbf{B}$ and α that are redundant wrt. $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$ are entailed by clauses in $(\Phi_\mathbf{B})^{\Lambda_\mathbf{B}, \alpha}$. The first part of the theorem, which is already proved, states, more formally, just $\mathcal{I}[\Lambda_\mathbf{B}, \pi, \alpha] \models (\Phi_\mathbf{B})^{\Lambda_\mathbf{B}, \alpha}$. Hence $\mathcal{I}[\Lambda_\mathbf{B}, \pi, \alpha] \models C \leftarrow c$. □

Theorem 7.1 (Refutational soundness) *Let* $\mathbf{D}$ *be a refutation from a set* Ψ *of input clauses and a satisfiable set* Γ *of closed constraints.*

If $\mathbf{T}$ *is a refutation tree in* $\mathbf{D}$*,* $\mathbf{B}$ *the rightmost branch in* $\mathbf{T}$*, and* $\Gamma_\mathbf{B}$ *the sequent in the leaf of* $\mathbf{B}$*, then* $\Gamma_\mathbf{B} \supseteq \Gamma$*,* $\Gamma_\mathbf{B}$ *is satisfiable, and* $\Gamma_\mathbf{B} \cup \Psi$ *is not satisfiable.*

Proof. Let $\mathbf{T}$, $\mathbf{B}$ and $\Gamma_\mathbf{B}$ as stated.

By structural induction on derivation trees we show a slight more general property:

(P) for every sequent $\Lambda \cdot \Gamma \vdash \Phi$ (labeling a node) in $\mathbf{T}$ there is a satisfiable set of background constraints Γ' with $\Gamma \subseteq \Gamma' \subseteq \Gamma_\mathbf{B}$ such that $\Lambda^{(e,n)} \cup \Phi^c \cup \Gamma'$ is not satisfiable.

From (P) follows immediately that $\Gamma_\mathbf{B} \cup \Psi$ is not satisfiable: in the initial sequent we have $\Lambda = \{\neg x\}$ and $\Phi^c = \Psi$. Thus, with (P), $\Psi^c \cup \Gamma'$ is not satisfiable, and with $\Gamma' \subseteq \Gamma_\mathbf{B}$, $\Psi^c \cup \Gamma_\mathbf{B}$ is not satisfiable either.

It remains to prove (*P*). Let $\Lambda \cdot \Gamma \vdash \Phi$ be a sequent in $\mathbf{T}$. If $\Lambda \cdot \Gamma \vdash \Phi$ is in a leaf node, then $\square \leftarrow \emptyset \cdot \top \in \Phi$ is the defining property of refutations, and setting $\Gamma' = \Gamma$ proves (P) immediately.

Otherwise assume that $\Lambda \cdot \Gamma \vdash \Phi$ has one or two successor nodes. We carry out a case analysis wrt. the inference rule applied to $\Lambda \cdot \Gamma \vdash \Phi$. We note first that every background context in every derived sequent in $\mathbf{T}$ is a subset of $\Gamma_\mathbf{B}$, the background context in the leaf of the rightmost branch in $\mathbf{T}$. This can be seen from inspection of the inference rules: in every Split inference the background context in the left conclusion is always a subset of the background context in the right conclusion, and none of the other (non-branching) inference rules shrinks background contexts. Hence, if we chose below Γ' to be a background constraint in one of the sequents in $\mathbf{T}$, $\Gamma' \subseteq \Gamma_\mathbf{B}$ will follow. Furthermore, by design of

the inference rules, satisfiability of background contexts is preserved for every background context in every derived sequent. It follows that$\Gamma_{\mathbf{B}}$ is satisfiable.

If the inference rule is Close then the selected clause is of the form $\Box \leftarrow R \cdot c$ and it holds $R \subseteq \Lambda$. The clause form of $\Box \leftarrow R \cdot c$ is $\overline{R} \leftarrow c$. Let $\overline{R}^{(e,n)}$ be obtained from $\overline{R}$ by replacing every variable by e and every rigid variable by d. With $R \subseteq \Lambda$ it follows that no interpretation satisfies $\Lambda^{(e,n)} \cup \{\Box \leftarrow R \cdot c\}^{\mathrm{c}}$. Finally, chose $\Gamma' = \Gamma \cup \{c\}$ and (P) follows immediately.

If the inference rule is Restrict, by induction (P) holds for its conclusion. Chosing Γ' as provided by the induction suffices to prove (P) for $\Lambda \cdot \Gamma \vdash \Phi$.

If the inference rule is Simp, by induction (P) holds for its conclusion. Again chosing Γ' as provided by the induction suffices to prove (P) for $\Lambda \cdot \Gamma \vdash \Phi$. Condition (2) in Simp is crucial here.

If the inference rule is Ref, Para or Pos-Res, let $C' \leftarrow R' \cdot c'$ be the derived clause. It is not difficult to show that each of these rules entails the derived clause in the sense $\Gamma \cup \Lambda^{(e,n)} \cup \Phi^{\mathrm{c}} \models (C' \leftarrow R' \cdot c')^{\mathrm{c}}$. Again the same argument as for Simp applies.

If the inference rule is Split let $\Lambda, \overline{K} \cdot \Gamma, c \vdash \Phi$ be the left conclusion and $\Lambda, K \cdot \Gamma^{\star} \vdash \Phi$ be the right conclusion. By induction applied to the left conclusion, there is a satisfiable set of background constraints Γ' with $\Gamma \cup \{c\} \subseteq \Gamma' \subseteq \Gamma_{\mathbf{B}}$ such that $(\Lambda \cup \{\overline{K})^{(e,n)} \cup \Phi^{\mathrm{c}} \cup \Gamma'$ is not satisfiable (*). By condition (4) in Split, $\Gamma^{\star}$ is any such set Γ'. As we have made no further assumption on Γ' we can set $\Gamma^{\star} = \Gamma'$.

By induction applied to the right conclusion, there is a satisfiable set of background constraints Γ'' with $\Gamma \cup \{c\} \subseteq \Gamma' \subseteq \Gamma'' \subseteq \Gamma_{\mathbf{B}}$ such that $(\Lambda \cup \{K\})^{(e,n)} \cup \Phi^{\mathrm{c}} \cup \Gamma''$ is not satisfiable. From (*) and with $\Gamma' \subseteq \Gamma''$ it follows that $(\Lambda \cup \{\overline{K})^{(e,n)} \cup \Phi^{\mathrm{c}} \cup \Gamma''$ is not satisfiable, either. These two sets differ only in the literal $K^{(e,n)}$ and $\overline{K}^{(e,n)}$. Because every interpretation satisfies a literal or its complement, it follows that $\Lambda^{(e,n)} \cup \Phi^{\mathrm{c}} \cup \Gamma''$ is not satisfiable. Finally choose Γ'' to complete the proof for (P) in the Split case. □