Model Evolution With Built-in Theories – Version 3

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

February 7, 2012

Abstract

Model Evolution is a lifted version of the propositional DPLL procedure for first-order logic with equality. This paper combines and extends the essentials of the latest Model Evolution variants with and without theory reasoning into a new calculus. The new calculus is described in detail. The main results reported here are the calculus' completeness under (unavoidable) conditions, and its application as a decison procedure for the quantifier-free fragment of the combined theory of free function symbols with equality and linear integer arithmetic.

1 Introduction

Many applications of automated deduction require reasoning modulo background theories, in particular some form of arithmetic. Developing sophisticated automated reasoning systems that are also able to deal with quantified formulas has recently been an active area of recent research. Many SMT [10] solvers, in particular those implementing the DPLL(T) approach, deal with quantfied formulas, however in an incomplete way that relies on heuristic instantiation [7]; complete instantiation exists for specific fragments [8]. Other approaches integrate theory reasoning into general first-order logic reasoning methods (with native quantifier support). Recent approaches of that kind are based on superposition [2, 9, 1], on sequent calculi [11], or on instance-based methods [6, 3, 5]. This paper contributes to this line of research and proposes a novel instance-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, "Version 3", combines the essentials of the latest non-theory Model Evolution (ME) calculus with equality, $\mathcal{ME}_{\mathrm{E}}$ [4], and the ME calculus extended with linear integer arithmetic, MELIA [3]. Version 3 improves previous work in this direction [5] by, among others, providing a much more powerful model representation. For example, the calculus in [5] does not terminate on the input $P(x) \leftarrow x \geq 0$, as it will enumerate $P(0), P(1), \ldots$ to represent the model of the formula. In Version 3, the model is represented in a finite way

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

by the constrained literal $P(x)|x \geq 0$. The general idea behind this model representation is taken from MELIA [3]. MELIA, however, does not support free function symbols (other than free constants of the background sort). Unlike the calculus in [5] Version 3 also supports free function symbols ranging into the background sort (the integers). Free constants of the background sort can be declared as such free function symbols or as symbolic constants of the background signature, called "parameters". The latter case often enables more efficient reasoning, but has higher demands on the background reasoner (quantifier elimination). For example, that a constant a has a domain $1 \leq a \leq 100$ can be expressed directly this way instead of using a clause $a \approx 1 \vee \cdots \vee a \approx 100$.

One main result in this paper is the refutational completeness of Version 3 in presence of semantically justified redundancy criteria and a generic simplification rule. The completeness result is conditional on certain criteria involving the free function symbols ranging into the background sort. Our second main result exploits this criteria and explains Version 3 as a decision procedure for the (important) quantifier-free fragment of the combined theory of free function symbols with equality and linear integer arithmetic. The result actually applies to a certain first-order logic extension of that fragment, which can be seen to lift Bernays-Schoenfinkel logic to the theory case, see Section 8. To my knowledge, none of the theorem-proving calculi above [2, 9, 9, 1, 11, 6] is known to achieve that. The results above lead one step further in the overall research plan underlying this work, to overcome the intrinsic limitations of current SMT approaches regarding quantified formulas, much like lifting the propositional DPLL procedure to the first-order level while preserving its good properties was the main motivation for Model Evolution.

As an example (without equality, for simplicity), the clause set consisting of (1) $P(x) \vee Q(x) \leftarrow x \geq a$ and (2) $\neg P(x) \leftarrow x \geq 3$, where a is a parameter, will be analysed as follows: first, (1) is satisfied by setting up a "foreground context" Λ (corresponding to a branch in a DPLL-like search tree) containing a context literal $P(x)|x \geq a$, which says "$P(x)$ is true for all $x \geq a$". However, clause (2) makes this impossible by "closing" Λ and fixing $a = 3$ in a "background context" Γ, which is a global data structure to accumulate constraints under which foreground contexts were closed. Backtracking then replaces $P(x)|x \geq a$ by $\neg P(x)|x \geq a$. This requires to work on (1) again, by adding $Q(x)|x \geq a$. The derivation stops here, and the model of (1) and (2) is described, essentially, by $Q(x)|x \geq a$ and $a = 3$.

2 Preliminaries

We assume usual notions 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. 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 complete first-order *background theory* T of signature Σ_{B} all of whose models interpret $=$ as the identity relation. The set $|\mathsf{B}|$ that T associates to the sort B is called the *background domain*. We assume that $|\mathsf{B}|$ is at most countably infinite and all of its elements are included in Σ as B-constant symbols.In this paper, T is fixed as the theory of linear integer arithmetic (LIA): We assume that 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 $|\mathsf{B}| = \{0, \pm 1, \pm 2, \ldots\}$. The calculus requires a well-ordering $\dot{\leq}$ on $|\mathsf{B}|$ of order type ω. For LIA, define $x \dot{\leq} y$ iff $|x| < |y|$ or ($|x| = |y|$ and $x \leq y$). The relations $x \doteq y$ iff $x \dot{\leq} y$ and $y \dot{\leq} x$, and $x \dot{<} y$ iff $x \dot{\leq} y$ and $x \neq y$ are derived as usual. This orders the integers as $0 \dot{<} -1 \dot{<} 1 \dot{<} -2 \dot{<} 2 \dot{<} -3 \dot{<} 3 \cdots$. Linear rational arithmetic can be dealt with in a similar (and practical) way.

The expanded signature $\Sigma_{\mathsf{B}}^{\Pi}$ is obtained by adding to Σ_{B} an infinite set Π of free constants of sort B called *parameters* (following common mathematical terminology). The function and predicate symbols of $\Sigma_{\mathsf{B}}^{\Pi}$ are collectively referred to as the *background operators*. The *full signature* Σ is obtained by adding to $\Sigma_{\mathsf{B}}^{\Pi}$ the foreground sort F, function symbols of given arities over B and F, and two infix equality predicate symbols, $\approx_{\mathsf{F}}$ and $\approx_{\mathsf{B}}$, of arities $\mathsf{F} \times \mathsf{F}$ and $\mathsf{B} \times \mathsf{B}$, respectively. We generally write $\approx$ as it causes no confusion. 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. For convenience, we often write non-equational atoms in examples.

Let X_{F} be an infinite set of F-*variables*, variables of sort F, disjoint from 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. The free B-sorted constants in these formulas can be declared as parameters or as (B-sorted) foreground constants. A *term* is a (well-sorted) $\Sigma(X)$-term, an *atom* is an equation $s \approx t$ between two $\Sigma(X)$-terms s and t. A *literal* is an equation or a negated equation, the latter also written as a disequation as in $s \not\approx t$. A *formula* is a (well-sorted) $\Sigma(X)$-formula. A *foreground term* is a term with no operators from $\Sigma_{\mathsf{B}}^{\Pi}$. Foreground terms are made of foreground symbols and variables; they can be B-sorted or F-sorted. 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) $\Sigma_{\mathsf{B}}^{\Pi}(X_{\mathsf{B}})$-term. A *ground term* is a term with no variables. A *pure ground term* is a ground term whose only background subterms are background domain elements. A *Herbrand term* is a ground term whose only B-sorted subterms are background domain elements. (Pure ground terms may contain B-sorted foreground function symbols, Herbrand terms not.) Intuitively, Herbrand terms do not contain symbols that need external evaluation, i.e., they contain no parameters, no variables, and no operators from Σ_{B} other than domain elements. For example, if f is B-sorted then $f(1,1)$ is pure, $f(1+1,1)$ is not, and none of them is a Herbrand term. If f is F-sorted then $f(1,1)$ and 1 are Herbrand terms, but $f(a,1)$ and a are not, if a is a background domain element, otherwise they are Herbrand terms as well.

If e is a term or a formula and $\vec{x}$ is a tuple of distinct variables we write $e[\vec{x}]$ to denote that e's free variables are all among $\vec{x}$. For a tuple $\vec{m}$ of terms of the same length, we write $e[\vec{m}]$ to denote the result from replacing the variables $\vec{x}$ in e by the terms $\vec{m}$.

A *substitution* is a mapping σ from variables to terms that is sort respecting, that is, maps each variable $x \in X$ to a term of the same sort. We write substitution application in postfix form, extend it to terms, formulas, sets of these, etc, in the expected ways. The *domain* of a substitution σ is the set $\mathrm{dom}(\sigma) = \{x \mid x \neq x\sigma\}$. We work with substitutions with finite domains only, written as $\sigma = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$ where $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$. A substitution is *simple* if every B-sorted variable in its domain is mapped to a background term. For example $[x \mapsto 1 + y + a]$ is simple, whereas $[x \mapsto f(1)]$ is not, if a is a parameter. A *Herbrand substitution* is a substitution that maps every variable to a Herbrand term. Notice that every Herbrand substitution is simple. We denote by *fvar*(F) the set of variables that occur free in F, where F is a term or formula. We assume a reduction ordering $\succ$ that is total on pure ground terms.[1] We also assume $\succ$ extends the ordering $\dot{<}$ on domain elements (i.e., $d_1 \succ d_2$ iff $d_2 \dot{<} d_1$) and that no non-domain element is smaller than any domain element. The ordering $\succ$ is extended to literals over pure terms by identifying a positive literal $s \approx t$ with the multiset $\{s, t\}$, a negative literal $s \not\approx t$ with $\{s, s, t, t\}$, and using the multiset extension of $\succ$. Multisets of literals are compared by the multiset extension of $\succ$, also denoted by $\succ$.

Definition 2.1 A *(T-based Herbrand) interpretation* I is any Σ-structure that (i) is identical to T over the symbols of Σ^{B}, (ii) interprets every F-sorted 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 Herbrand terms from the proper domain, (iii) interprets every B-sorted foreground n-ary function symbol f as an n-ary total function f^I from the proper domains to $|\mathsf{B}|$, and (iv) interprets $\approx$ as a congruence relation on F-sorted Herbrand terms.[2] □

If in item (iii) "total" is replaced by "partial" we speak of *partial (T-based Herbrand) interpretations*. A *(parameter) valuation* π is a mapping from Π to $|\mathsf{B}|$. Since all the elements of $|\mathsf{B}|$ are constants of Σ_{B} we will often treat valuations similarly to substitutions. For any Herbrand interpretation I and valuation π, 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. The symbols I and π will always denote respectively Herbrand interpretations and valuations. Hence, we will often use the symbols directly, without further qualification. We use, possibly with subscripts, the letters $\{x, y\}$, $\{a, b\}$, and $\{f, g\}$ to denote respectively variables, parameters, and foreground function symbols.

Constraints. We call *(background) constraint* any formula in the closure of the set of $\Sigma^{\Pi}_{\mathrm{B}}(X_{\mathrm{B}})$-atoms under conjunction, negation and existential quantification of variables. A *closed constraint* is a constraint with no free variables. We denote by $\bar{\exists}\, c$ (resp. $\bar{\forall}\, c$) the existential (resp. universal) closure of the constraint c, and by $\pi\ \vec{x}\ c$ the *projection*

[1] 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 $s\sigma \succ t\sigma$.

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

of c on $\vec{x}$, i.e., $\exists \vec{y}\, c$ where $\vec{y}$ is a tuple of all the free variables of c that are not in the variable tuple $\vec{x}$. For example, if $c = x + y > 0$ then $\pi\, x\, c = \exists y\ x + y > 0$.

Definition 2.2 (Satisfaction of constraints) Let c be a closed constraint. We say that π *satisfies* c, written as $\pi \models c$, if $T \models c\pi$ in the standard sense.[3] □

A possibly non-closed constraint c is *π-satisfiable* iff $\pi \models \bar{\exists}\, c$. Two constraints c and d are *equivalent*, written as $c \equiv d$, iff every valuation satisfies $\bar{\forall}(c \leftrightarrow d)$. For a set Γ of closed constraints, say that π *satisfies* Γ, or that Γ is *π-satisfied*, written $\pi \models \Gamma$, iff π satisfies every $c \in \Gamma$. The set Γ is *satisfiable* iff some π satisfies it. By $\mathrm{mods}(\Gamma)$ denote the set of all valuations π such that $\pi \models \Gamma$. Since constraints contain no foreground symbols, for any interpretation $I[\pi]$ and closed constraint c, $I[\pi] \models c$ iff $\pi \models c$.

For any valuation π, a tuple $\vec{m}$ of integer constants is a *π-solution* of a constraint $c[\vec{x}]$ if $\pi \models c[\vec{m}]$. For instance, $[a \mapsto 3] \models c[4, 1]$, where $c[x, y] = (a = x - y)$.

We use the predicate symbol $\dot{\leq}$ also to denote the component-wise extension of the integer well-ordering $\dot{\leq}$ to integer tuples (for any tuple size), $\dot{\leq}_\ell$ to denote the lexicographic extension of $\dot{\leq}$ to integer tuples, and $\dot{<}$ and $\dot{<}_\ell$ to denote their respective strict version.[4] The calculus works with constraints that denote minimal π-solutions $m_1, ..., m_n$ of a π-satisfiable constraint $c[\vec{x}]$. This can be done with the operator μ_k defined below. We need further operators, $\dot{<}_\mu$ for comparing minimal π-solutions of constraints, and μ^1 for getting the unique π-solution of a constraint if it exists, also defined below. In that, $\vec{y}$ is a tuple of fresh variables with the same length as $\vec{x}$ and $k \geq 1$.

$$\mu\, c \stackrel{\text{def}}{=} c \wedge (\forall \vec{y}\, c[\vec{y}] \rightarrow \neg(\vec{y} \dot{<} \vec{x})) \qquad \mu_\ell\, c \stackrel{\text{def}}{=} c \wedge \forall \vec{y}\, (c[\vec{y}] \rightarrow \vec{x} \dot{\leq}_\ell \vec{y})$$

$$\mu_k\, c \stackrel{\text{def}}{=} \mu_\ell\, (\neg(\mu_1\, c) \wedge \cdots \wedge \neg(\mu_{k-1}\, c) \wedge (\mu\, c)) \qquad \mu^1\ c \stackrel{\text{def}}{=} c \wedge \forall \vec{y}\, (c[y] \rightarrow \vec{x} \doteq \vec{y})$$

$$c \dot{<}_\mu d \stackrel{\text{def}}{=} \forall \vec{x}\, \forall \vec{y}\, (\mu\, c[\vec{x}] \wedge \mu\, d[\vec{y}] \rightarrow \vec{x} \dot{<} \vec{y})$$

For example, $\mu\ (x < -5)$ has a unique minimal π-solution, which is -6, for all π. The constraint $c \dot{<}_\mu d$ expresses that *all* minimal π-solutions of c are strictly smaller than *all* minimal π-solutions of d. For example, if $c = x + y \geq 1$ and $d = x > y$ then c's minimal solutions are $(0, 1)$ and $(1, 0)$ and d's minimal solution is $(1, 0)$. Because of the overlap, $c \dot{<}_\mu d$ is not satisfied.

3 Contexts

A *context literal* is a pair $L|c$ where L is a foreground literal and c is a constraint.[5] Context literals are *normalized* by (i) purifying L, i.e., extracting every non-variable background term from L and conjoining the proper equation to c, then (ii) adding

[3] Observe that the test $T \models c\pi$ is well formed because $c\pi$ is closed and contains no parameters.

[4] We remark that each of the new symbols is definable in the given constraint language, see [3].

[5] Context literals in this paper are always "parametric", for simplicity, "universal" ones will be added later. See [4] for a (non-theory) ME calculus that has both.

constraints $0 \dot{\leq} x$ to that c for every B-sorted variable x in L that does not occur free in that c, and finally (iii) adding existential quantifications to that c for the free variables in that c that do not occur in L. The purpose of step (ii) is to commit the mentioned variables x to the number 0 for the purpose of closing branches, see Definition 5.1 below. With 0 being the least element in the ordering $\dot{\leq}$, step (ii) does does not change the mening of context literals ("productivity", see below) in any way .

We need to consider only context literals whose normalization removes all background operators from L. For example, $P(a+1)|\top$ could not be normalized if a is a foreground symbol. We assume all context literals are always normalized, and non-normalized context literals in examples always denote their normalized version. For example, $P(x, y{+}1, u, v)|(x{+}z = 0)$ denotes $P(x, x', u, v)|(\exists yz\ x{+}z = 0 \wedge x' = y{+}1 \wedge 0 \dot{\leq} u)$ (v is F-sorted). An ordinary literal L is taken as the normalized version of the context literal $L|\top$ if the context demands. The application of a simple substitution δ to a constrained literal $L|c$, written as $(L|c)\delta$, is the normalized version of the constrained literal $L\delta|c\delta$. For any context literal K and renaming substitution ρ we call $K\rho$ a *variant* of K. The *complement* $\overline{L|c}$ of a context literal is defined as $\overline{L}|c$, where $\overline{L}$ is the usual complement operation on literals.

A *foreground context* Λ is a sequence of context literals. We assume that every foreground context starts with a pseudo-literal $\neg x|\top$, even when not explicitly written. We will take Λ also as a set of context literals that is closed under adding variants of its elements. This allows us, e.g., to write $K \in \Lambda$ instead of the more clumsy "$L \in \Lambda$, for some variant L of K". A *background context* Γ is a set of closed constraints. A *context* is a pair $\Lambda \cdot \Gamma$ consisting of a foreground context Λ and a background context Γ.

For constraints c and d, we define a preorder parametrized in a valuation π as $c \gtrsim^\pi d$ iff $\pi \models \bar{\forall}(d \to c)$ and d is π-satisfiable. (Intuitively, $c \gtrsim^\pi d$ says that the extension of c includes that of d, hence "$\gtrsim$ "). The preorder $\gtrsim^\pi$ induces an equivalence relation $\sim^\pi$ as usual, $c \sim^\pi d$ iff $c \gtrsim^\pi d$ and $c \gtrsim^\pi d$, and its strict subset, $c \gtrsim^\pi d$ iff $c \gtrsim^\pi d$ and $c \not\gtrsim^\pi d$.

We need to compare context literals wrt the instances they denote. Let $K_1 = L_1|c_1$ and $K_2 = L_2|c_2$ be context literals. We say that K_1 *π-covers* K_2, written as $K_1 \gtrsim^\pi K_2$, iff there is a substitution σ such that $L_1\sigma = L_2$ and $c_1\sigma \gtrsim^\pi c_2$. In this case we also call K_2 a *π-instance* of K_1. Notice that σ above will always be simple. This is a consequence of assuming that context literals are normalized. Hence, $c_1\sigma$ is always a formula over the background signature, i.e., a constraint, and asking whether $c_1\sigma \gtrsim^\pi c_2$ holds is well-defined. The equivalence relation and the strict subset induced by the π-covering relation are defined analogously to above on constraints. They are also denoted by the symbols $\sim^\pi$ and $\gtrsim^\pi$, respectively. For example $(P(x, y)|3 \leq x) \gtrsim^\pi (P(x, g)|a \leq x)$ if $\pi = [a \mapsto 4]$ as per $\sigma = [y \mapsto g]$ and $\pi \models \forall x\ (a \leq x \to 3 \leq x)$. The following definition is a first application of these concepts.

Definition 3.1 (π-contradictory) A context literal K is *π-contradictory with Λ* iff there is a $K' \in \Lambda$ such that $K' \sim^\pi \overline{K}$. □

The context Λ is π-contradictory iff it contains a context literal that is π-contradictory with Λ. The calculus works with a constraint representation of Definition 3.1. The

set $-\text{contra}(L|c, \Lambda) \stackrel{\text{def}}{=} \{\neg\bar{\forall}(c \leftrightarrow d) \mid \overline{L}|d \in \Lambda\}$ provides constraints on valuations π under which a given context literal is *not* π-contradictory with Λ. For example, if $K = P(x)|x = a$ and $\Lambda = \{\neg P(x)|x = a \wedge a = 0, \neg P(x)|x = a \wedge a > 0\}$ then $-\text{contra}(K, \Lambda) = \{\neg\forall x(x = a \leftrightarrow x = a \wedge a = 0), \neg\forall x(x = a \leftrightarrow x = a \wedge a > 0)\}$, simplified $\{a \neq 0, a \leq 0\}$, which is π-satisfied by all π such that $a\pi < 0$, as expected. Context that are π-contradictory must be avoided. Intuitively, they are not well-defined for specifying truth values, a notion derived from the concept of "productivity" introduced below.

The strict ordering $\dot{<}_\mu$ introduced earlier is extended to context literals as follows. Let $K_1 = L_1|c_1$ and $K_2 = L_2|c_2$ be context literals. We say that *K_1 is strictly more π-specific than K_2*, written as $K_1 \dot{<}^\pi_\mu K_2$, iff there is a substitution σ such that $L_1 = L_2\sigma$ and if L_1 and L_2 are variants then $\pi \models c_1 \dot{<}_\mu c_2\sigma$. $K_1 \dot{<}^\pi_\mu K_2$ means that L_1 is a proper instance of L_2 or L_1 and L_2 are variants and the minimal π-solutions break the tie.

The following definition provides the basis for determinimg whether a (ground) literal is true in the interpretation induced by a context.

Definition 3.2 (Productivity) Let K and L be context literals. We say that *K π-produces L in Λ* iff (i) $K \gtrsim^\pi L$ and (ii) there is no $K' \in \Lambda$ such that $\overline{K'} \dot{<}^\pi_\mu K$ and $\overline{K'} \gtrsim^\pi L$. □

We say that Λ *π-produces L* iff some $K \in \Lambda$ π-produces L in Λ. As a simple example, $\Lambda = \{P(x, g_2), \neg P(g_1, y}\}$ π-produces both $P(g_1, g_2)$ and $\neg P(g_1, g_2)$, for all π. After adding $P(g_1, g_2)$ to Λ, Λ does no longer produce $\neg P(g_1, g_2)$.

Given Λ and L, thanks to the strict ordering $\dot{<}^\pi_\mu$, every sequence of context literals $(K)_{i\in I}$ such that $K_{i+1} \dot{<}^\pi_\mu K_i$ and $K_i \gtrsim^\pi L$ in Λ must be finite. This is an important detail to ensure that every context Λ π-produces L or $\overline{L}$, even if Λ is infinite.

4 Constrained Clauses

A *(constrained) clause* is an expressions of the form $C \leftarrow R \cdot c$ where R is a multiset of foreground literals, the set of *context restrictions*, c is a constraint, and C is an ordinary foreground clause. When C is empty we write it as $\square$. When R is empty, we write the constrained clause more simply as $C \leftarrow c$ and call it *input (constrained) clause*.

Any expression of the form $C \leftarrow c$ where C is an arbitrary ordinary Σ-clause and c a constraint can be turned into an input clause by abstracting out offending subterms from C and moving them to the constraint side of $\leftarrow$. For example, $P(a, v, x + 5) \leftarrow x > v$ becomes $P(x_1, v, x_2) \leftarrow x > v \wedge x_1 = a \wedge x_2 = x + 5$. We assume every constrained clause $C \leftarrow R \cdot c$ is *normalized*, that is, C does not contain any (dis)equation between B-sorted variables. This can be achieved by removing all offending equations $x \approx y$ ($x \not\approx y$) from C, where x and y are B-sorted, and conjoining their complements $x \neq y$ ($x = y$) to c. As will be clear later, all the transformations above preserve the semantics of the original expression, and, except normalization, can be done once and for all prior to derivations.

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.

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 $C \leftarrow c$. A Herbrand instance $C \leftarrow c$ can be evaluated directly by a (total) interpretation I: we say that *I satisfies $C \leftarrow c$*, written $I \models C \leftarrow c$ if $I \models C \vee \neg c$. For input clauses $C \leftarrow c$ we say that *I satisfies $C \leftarrow c$* iff I satisfies every Herbrand instance of $C \leftarrow c$. For a set Δ of input clauses or closed constraints we say that *I satisfies Δ*, written as $I \models \Delta$, if $I \models \Delta$, for every $F \in \Delta$. We say that *Δ is satisfiable* if some I satisfies Δ. Let G be an input clause or closed constraint. We say that *Δ entails G*, written as $\Delta \models G$, if every interpretation I that satisfies Δ also satisfies G.

The definition of satisfaction of general constrained 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 $R \cdot c$ is evaluated with respect to productivity in a context. Moreover, constrained clause satisfaction is not definable purely at the ground level and requires *(Herbrand) closures*, that is, pairs of the form $(C \leftarrow R \cdot c, \gamma)$ where γ is a Herbrand substitution.

Definition 4.1 (Satisfaction of context restrictions) The pair *(Λ, π) satisfies $(R \cdot c, \gamma)$*, written as $(\Lambda, \pi) \models (R \cdot c, \gamma)$, if $\pi \models c\gamma$ and

(i) $R\gamma$ contains no *trivial literals*, of the form $t \approx t$ or $\neg(t \approx t)$, and for every $l \approx r \in R$, if $l\gamma \succ r\gamma$ then l is not a variable, and

(ii) for every $L \in R$ there is a $K \in \Lambda$ that π-produces both $L|c$ and $L\gamma|c\gamma$ in Λ.

□

Point (i) makes paramodulation into variables unnecessary for completeness in the calculus. For example $(\{P(x)|x \geq 2\}, \emptyset) \not\models (\{P(x), P(y)\} \cdot x \neq y \wedge x \geq 2 \wedge y \geq 2, [x \mapsto 2, y \mapsto 2])$ as although $P(x)|x \geq 2$ $\emptyset$-produces $P(x)|\exists y\ x \neq y \wedge x \geq 2 \wedge y \geq 2$ in $\{P(x)|x \geq 0\}$ but $P(x)|x \geq 2$ does not $\emptyset$-produce $P(2)|2 \neq 2 \wedge 2 \geq 2 \wedge 2 \geq 2$ in $\{P(x)|x \geq 0\}$. Similarly for $P(y) \in R$. Notice that the above definition does *not* check whether K π-produces $(L|c)\gamma$ in Λ, which could be different due to implicit normalization. Indeed, in the example, $(L|c)\gamma = (P(x)|\exists y\ x \neq y \wedge x \geq 2 \wedge y \geq 2)\gamma = P(2)|\exists y\ 2 \neq y \wedge 2 \geq 2 \wedge y \geq 2$.

Definition 4.2 (Satisfaction of Herbrand closures) A triple *(Λ, π, I) satisfies $(C \leftarrow R \cdot c, \gamma)$*, written as $(\Lambda, \pi, I) \models (C \leftarrow R \cdot c, \gamma)$, iff $(\Lambda, \pi) \not\models (R \cdot c, \gamma)$ or $I \models (C \leftarrow c)\gamma$. □

We will use Definition 4.2 always with $I = I[\pi]$. 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, $(\Lambda, \pi, I[\pi]) \models (C \leftarrow \emptyset \cdot c, \gamma)$ if and only if $I[\pi] \models (C \leftarrow c)\gamma$.

5 Core Inference Rules

The calculus works on sequents of the form $\Lambda \cdot \Gamma \vdash \Phi$, also called *states* (together with a distinguished "Fail" state), where $\Lambda \cdot \Gamma$ is a context and Φ is a set of constrained clauses. Below we introduce its inference rules as state transition operators $\Longrightarrow$, as customarily used for SMT procedures [10], and with similar names. We also use the symbol $\Longrightarrow$ to denote the transition relation defined by the operators and call its elements *inferences*.

There are eight *core inference rules*: Ref, Para, Neg-Res, Decide-1, Decide-2, and Minsol, the *evolution rules*, and Fail and Backtrack, the *closing rules*. In their description, if S is a set and a is an element, we will write S, a as an abbreviation of $S \cup \{a\}$. Each one operates on a clause as specified in its description, which is referred to as the *selected clause* further below. The concatenation of two foreground contexts will be denoted by simple juxtaposition. Context literals can be labeled as *decision literals*, which is indicated with the superscript $\cdot^d$. We assume that every clause with a non-empty ordinary clause part has at least one of its literals *selected*.

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

$$\Lambda \cdot \Gamma \vdash \Phi \quad \Longrightarrow \quad \Lambda \cdot \Gamma \vdash \Phi, (C \leftarrow R \cdot c)\sigma \tag{Ref}$$

if Φ contains a clause $s \not\approx t \vee C \leftarrow R \cdot c$ with selected literal $s \not\approx t$ such that σ is a simple mgu of s and t.

The next inference rule is a variant of ordered paramodulation.

$$\Lambda \cdot \Gamma \vdash \Phi \quad \Longrightarrow \quad \Lambda \cdot \Gamma \vdash \Phi, (L[r] \vee C \leftarrow (R \cup \{l \approx r\}) \cdot c \wedge d)\sigma \tag{Para}$$

if Φ contains a clause $L[s] \vee C \leftarrow R \cdot c$ with selected literal $L[s]$, and Λ contains a literal $l \approx r|d$ such that (i) σ is a simple mgu of l and s, (ii) s is not a variable, (iii) $r\sigma \not\succeq l\sigma$, and (iv) if $L[s]$ is of the form $t[s] \not\approx t'$ and s does not start with a B-sorted foreground function symbol then $t'\sigma \not\succeq t[s]\sigma$.

Notice that the resulting clause may require normalization, precisely when $L[r]$ is a (dis)equation between B-sorted variables. The context literal $l \approx r|d$ is assumed to be variable disjoint with the mentioned clause. It is added to $R \cdot c$ to preserve soundness.

$$\Lambda \cdot \Gamma \vdash \Phi \quad \Longrightarrow \quad \Lambda \cdot \Gamma \vdash \Phi, (C \leftarrow R \cup \{s \not\approx t\} \cdot c \wedge d)\sigma \tag{Neg-Res}$$

if Φ contains a clause of the form $s \approx t \vee C \leftarrow R \cdot c$ with selected literal $s \approx t$ and Λ contains a context literal of the form $\neg A|d$ such that σ is a simple mgu of A and $s \approx t$. (The literal $\neg A|d$ could be the pseudo-literal $\neg x|\top$.)

In its core, an inference with one of the above three rules takes the selected clause D and a context literal K (except Ref) to derive a new clause. If γ is a Herbrand substitution such that the applicability conditions stated with that rule holds for $D\gamma$, $K\gamma$ and $\sigma = \epsilon$, we say that the inference *admits a ground instance via* γ.

Inference rules for extending contexts. The rules below intend to modify the current context $\Lambda \cdot \Gamma$ to satisfy a currently falsified empty clause $\square \leftarrow R \cdot c$. Intuitively, this is the case only if for some ground instance $\square \leftarrow R' \cdot c'$ of $\square \leftarrow R \cdot c$ and some valuation $\pi \in \mathrm{mods}(\Gamma)$, c' is π-satisfied and Λ produces $L'|c'$ for all $L' \in R'$. The Decide rules then "repair" this situation by adding to Λ a *decision literal* whose effect is that afterwards $L'|c'$ is no longer produced, hence $\square \leftarrow R' \cdot c'$ is satisfied.

$$\Lambda \cdot \Gamma \vdash \Phi \quad \Longrightarrow \quad \Lambda\ (\overline{L}|c)^d \cdot \Gamma \cup \Gamma' \vdash \Phi \qquad \text{(Decide-1)}$$

if Φ contains a clause D of the form $\square \leftarrow R \cup L \cdot c$ and Λ contains a context literal K such that (i) $K \gtrsim^{\pi} L|c$, for some $\pi \in \mathrm{mods}(\Gamma)$, (ii) $\Gamma' = -\mathrm{contra}(\overline{L}|c, \Lambda)$, and (iii) $\Gamma \cup \Gamma'$ is satisfiable.

Informally, the Decide-1 rule considers the case that K π-produces $L|c$ in Λ and L is a proper instance of the literal of K (this aspect is not checked by the rule, though). Extending the background context with Γ' makes sure that the new foreground context is not π-contradictory, for no $\pi \in \mathrm{mods}(\Gamma \cup \Gamma')$.

$$\Lambda \cdot \Gamma \vdash \Phi \quad \Longrightarrow \quad \Lambda\ (\overline{L}|c \wedge \neg \mu d)^d \cdot \Gamma \cup \Gamma' \vdash \Phi \qquad \text{(Decide-2)}$$

if Φ contains a clause D of the form $\square \leftarrow R \cup L \cdot c$ and Λ contains a variant of the context literal of the form $L|d$ such that (i) $\Gamma' = -\mathrm{contra}(\overline{L}|c \wedge \neg \mu d, \Lambda)$, and (ii) $\Gamma \cup \Gamma'$ is satisfiable.

The Decide-2 rule complements Decide-1 by considering the case that $L|d$ π-produces $L|c$ in Λ. To achieve the desired effect requires to remove all minimal π-solutions of d from the decision literal, hence the constraint $c \wedge \neg \mu d$.

$$\Lambda \cdot \Gamma \vdash \Phi \quad \Longrightarrow \quad \Lambda\ (L|\mu_k d)^d \cdot \Gamma \cup \Gamma' \vdash \Phi \qquad \text{(Minsol)}$$

if Φ contains a clause D of the form $\square \leftarrow R \cup L \cdot c$ and Λ contains a context literal of the form $L|d$ such that (i) $\Gamma' = -\mathrm{contra}(\overline{L}|\mu_k d, \Lambda) \cup -\mathrm{contra}(L|\mu_k d, \Lambda)$, and (ii) $\Gamma \cup \Gamma'$ is satisfiable.

The Minsol rule adds minimal π-solution instances of a context literal. They can be used by the closing rules below. Conditions (i) and (ii) correspond in the propositional case to not adding a literal L to a context if the context contains already L or $\overline{L}$. Definition 6.3 provides criteria to control Minsol.

Inference rules for closing contexts. Intuitively, closing a context means abandoning it because a ground instance of some constrained empty clause is irrepairably falsified. This is done on the first-order level by pairing the restriction literals in the clause with complementary context literals with *unique* π-solutions (for soundness reasons).

Definition 5.1 (Closing constraint) Let Λ be a context and $D = \square \leftarrow L_1, \ldots, L_n \cdot c$ a constrained empty clause. A constraint e is a *closing constraint for D and Λ* iff there are context literals $L_1|c_1, \ldots, L_n|c_n \in \Lambda$ such that $e \equiv \bar{\exists}\ (c \wedge \mu^1 c_1 \wedge \cdots \wedge \mu^1 c_n)$. $\square$

Each of the literals $L_i|c_i$ is said to *participate* in e. A sequent $\Lambda \cdot \Gamma \vdash \Phi$ with a clause $D \in \Phi$ is *closable (with D)* if there is a sequent $\Lambda' \cdot \Gamma' \vdash \Phi'$ that can be obtained from $\Lambda \cdot \Gamma \vdash \Phi$ by zero or more Minsol inferences such that there is a closing constraint e for D and Λ' and $\Gamma' \cup \{e\}$ is satisfiable.

If a sequent is closable, it will be abandoned in a derivation by means of Fail or Backtrack defined below(unless one of the Minsol inferences becomes non-applicable). In case of Backtrack a different part of the search tree will be explored by considering the complement of the most recent decision literal, as in propositional DPLL. (Section 7 below introduces derivations and search trees.) Notice that decision literals can participate in closing constraints only if they admit *unique* solutions. This guarantees that only the *same* solution can be used for the complementary literal to participate in a closing context unifier, which is essential to get a sound calculus.

$$\Lambda \cdot \Gamma \vdash \Phi, (\Box \leftarrow R \cdot c) \quad \Longrightarrow \quad \text{Fail with } \Gamma \cup \{e\} \qquad \text{(Fail)}$$

if (i) e is a closing constraint for $\Box \leftarrow R \cdot c$ and Λ, (ii) $\Gamma \cup \{e\}$ is satisfiable, and (iii) Λ contains no decision literal.

The result "Fail with $\Gamma \cup \{e\}$" provide constraints on the parameter valuations under which the input clause set is *not* π-satisfiable.

$$\Lambda_1 \; K^d \; \Lambda_2 \cdot \Gamma \vdash \Phi, (\Box \leftarrow R \cdot c) \quad \Longrightarrow \quad \Lambda_1 \; \overline{K} \cdot \Gamma \cup \{e\} \vdash \Phi_1 \qquad \text{(Backtrack)}$$

if (i) e is a closing constraint for $\Box \leftarrow R \cdot c$ and Λ , (ii) $\Gamma \cup \{e\}$ is satisfiable, (iii) Λ_2 contains no decision literal, and (iv) $\Lambda_1 \cdot \Gamma_1 \vdash \Phi_1$ is a previous state, for some Γ_1. Notice that $\overline{K}$ is not labeled as a decision literal.

6 Model construction, redundancy and static completeness

In this section we show how to derive from a sequent $\Lambda \cdot \Gamma \vdash \Phi$ an intended interpretation $I[\Lambda, \pi]$ as a canonical candidate model for Φ, for each $\pi \in \text{mods}(\Gamma)$.

A *rewrite rule* is an expression of the form $l \to r$ where l and r are pure terms of the same sort. It is *F-sorted (B-sorted)* if l (and hence r) are F-sorted (B-sorted). A B-sorted rule is *suitable (for specifying interpretations)* iff it is of the form $f(t_1, \ldots, t_n) \to t_{n+1}$ where all t_i's are Herbrand terms and f is an n-ary B-sorted foreground operator. A F-sorted rule is *suitable* if both l and r and Herbrand terms. A *rewrite system* is a set of rewrite rules. The rewrite systems constructed below will be ordered, that is, consist of (suitable) rules of the form $l \to r$ such that $l \succ r$. For a given Λ and $\pi \in \text{mods}(\Gamma)$, we define by induction on the term ordering $\succ$ sets ϵ_K and $\mathcal{R}_K$ for every ground equation K between pure terms. Assume that ϵ_L has already been defined for all such L with $K \succ L$. Let $\mathcal{R}_K = \bigcup_{K \succ L} \epsilon_L$, where

$$\epsilon_{l \approx r} = \begin{cases} \{l \to r\} & \text{if } \Lambda \ \pi\text{-produces } l \approx r,\ l \succ r,\ l \to r \text{ is suitable and } l \\ & \text{and } r \text{ are irreducible wrt } \mathcal{R}_{l \approx r} \\ \emptyset & \text{otherwise} \end{cases}$$

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

For example, let $\Lambda = \{f(x) = x | 0 \leq x, f(x) = y | x = 1 \wedge y = 0, \neg f(x) = x | a \leq x\}$ and $\pi = [a \mapsto 3]$. Then $\mathcal{R}_{\Lambda,\pi}$ contains $f(0) \to 0$, $f(2) \to 2$ and $f(1) \to 0$. But it does not contain $f(1) \to 1$, which, although π-produced, is reducible by the smaller rule $f(1) \to 0$. It contains no $f(i) \to i$, for no $i \geq 3$, as the third literal in Λ prevents Λ from π-producing these. In summary, Λ and π specifies the partial interpretation for f via the rules $f(0) \to 0$, $f(2) \to 2$ and $f(1) \to 0$.

Definition 6.1 The *partial interpretation induced by Λ and π*, written as $I[\Lambda, \pi]$, is the Herbrand interpretation $I[\pi]$ that interprets foreground equality as the congruence closure of the F-sorted rules in $\mathcal{R}_{\Lambda,\pi}$ taken as equations, and that interprets partially the B-sorted foreground operators as specified by the B-sorted rules in $\mathcal{R}_{\Lambda,\pi}$. □

The rewrite system $\mathcal{R}_{\Lambda,\pi}$ is fully reduced by construction (no rule in $\mathcal{R}_{\Lambda,\pi}$ rewrites any other rule in it). Since $\succ$ is well-founded on the pure ground terms, $\mathcal{R}_{\Lambda,\pi}$ is convergent. It follows from well-known results that equality of F-sorted Herbrand terms in $I[\Lambda, \pi]$ can be decided by reduction to normal form using the F-sorted rules in $\mathcal{R}_{\Lambda,\pi}$. Regarding the B-sorted foreground operators, the restriction to *suitable* rules guarantees that the induced partial interpretation is well-defined. A rule like $f(g) \to 1$, where g is an (irreducible) B-sorted foreground operator will be ignored.

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 this ordering is in general not total (not only because constraints are ignored) but it is clear it is well-founded. This still suffices to identify minimal closures for the completeness proof, albeit not least ones.

Definition 6.2 (Redundant closure) Let $\Lambda \cdot \Gamma \vdash \Phi$ be a state and D a closure. We say that a closure $(C \leftarrow R \cdot c, \gamma)$ is *redundant wrt $\Lambda \cdot \Gamma \vdash \Phi$ and D* iff (a) $(\Lambda, \pi) \not\models (R \cdot c, \gamma)$ for all $\pi \in \text{mods}(\Gamma)$, or (b) there exist closures $(C_i \leftarrow R_i \cdot c_i, \gamma_i)$ of clauses in Φ, where $1 \leq i \leq n$ for some $n \geq 0$, such that for all $\pi \in \text{mods}(\Gamma)$

(i) for every $L \in R_i$ there is a $K \in R$ such that $L|c_i \sim^\pi K|c$ and $L\gamma_i|c_i\gamma_i \sim^\pi K\gamma|c\gamma$,

(ii) $\pi \models c\gamma \to c_i\gamma_i$,

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

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

□

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

If case (a) in the previous definition applies then $(C \leftarrow R \cdot c, \gamma)$ is trivially satisfied by $(\Lambda, \pi, I[\pi])$, for every $\pi \in \text{mods}(\Gamma)$ and every $I[\pi]$. Case (b) provides with (ii) and

(iv) conditions under which $(C \leftarrow c)\gamma$ follows from the $(C_i \leftarrow c_i)\gamma_i$'s. The context restrictions are taken into account by condition (i), which makes sure that evaluation of the pairs $(R_i \cdot c_i, \gamma_i)$ in terms of Definition 4.1 is the same as for $(R \cdot c, \gamma)$. In condition (iv), entailment $\models$ is meant as entailment in equational clause logic between sets of ordinary ground clauses and an ordinary ground clause.

Definition 6.3 (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 D is *redundant* wrt $\Lambda' \cdot \Gamma' \vdash \Phi'$ iff (a) for every Herbrand substitution γ, (D, γ) is redundant wrt $\Lambda' \cdot \Gamma' \vdash \Phi'$ or (b) the following holds, depending on the inference rule applied:

Ref, Para, Neg-Res: the inference does not admit a ground instance via γ, or (D', γ) is redundant wrt (D, γ) and $\Lambda' \cdot \Gamma' \vdash \Phi'$, where D' is the derived clause of that inference.

Decide-1, Decide-2: An inference with premise $\Lambda' \cdot \Gamma' \vdash \Phi'$, same selected clause D and same decision literal K does not exist (because the applicability conditions are not met), or Λ' does not π-produce $\overline{K}$, for no $\pi \in \mathrm{mods}(\Gamma')$

Minsol: An inference with premise $\Lambda' \cdot \Gamma' \vdash \Phi'$, same selected clause D and same committed literal K does not exist, or there is no closing constraint e for D and Λ' such that $\Gamma' \cup e$ is satisfiable and K participates in e.

□

We note that actually carrying out an inference makes it redundant wrt. its conclusion. The Decide and Minsol rules split the search space and go down the, say, left branch. After Backtrack into the right branch the inference does no longer exist, as the decision literal will be π-contradictory for all π satisfying the background context there (background context grow monotonically in derivations).

Definition 6.4 (Saturated sequent) A sequent $\Lambda \cdot \Gamma \vdash \Phi$ is *saturated* iff every inference with an evolution rule with premise $\Lambda \cdot \Gamma \vdash \Phi$ is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$. □

We note that Definitions 6.3 and 6.4 apply in a meaningful way also to sequents with infinite components. The Backtrack rule, however, makes sense only when the foreground context is finite. The static completeness result below holds only for saturated sequents with respect to *relevant* closures.

Definition 6.5 (Relevant closure) We say that a closure $(C \leftarrow R \cdot c, \gamma)$ is *relevant wrt. Λ and π* iff (i) for every $l \approx r \in R\gamma$, if $l \succ r$ then $l \to r \in \mathcal{R}_{\Lambda,\pi}$, and (ii) for every $l \not\approx r \in R\gamma$, if $l \succ r$ then $l \to r$ is suitable, and l and r are irreducible wrt. $\mathcal{R}_{\Lambda,\pi}$. □

Notice that all Herbrand closures of input clauses are always relevant.

In general it is impossible to derive sequents Λ that provide with $\mathcal{R}_{\Lambda,\pi}$ total interpretations for the B-sorted foreground operators, not even in the limit of derivations. (That would solve a Π^1_1-hard problem.) To work around that, we say that a sequent

$\Lambda \cdot \Gamma \vdash \Phi$ is *sufficiently complete* iff for every clause $C \leftarrow R \cdot c \in \Phi$ and every closure $(C \leftarrow R \cdot c, \gamma)$ that is not redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ it holds that $C\gamma$ is reducible by $\mathcal{R}_{\Lambda,\pi}$ to a clause without any B-sorted foreground operators. This suffices to remove all B-sorted foreground function symbols when it is theoretically needed.

Theorem 6.6 (Static completeness) *Let $\Lambda \cdot \Gamma \vdash \Phi$ be a saturated sequent that is not closable and π a valuation. If $\pi \models \Gamma$ and $\Lambda \cdot \Gamma \vdash \Phi$ is sufficiently complete, then every total extension of $I[\Lambda, \pi]$ satisfies all Herbrand closures of all clauses in Φ that are relevant wrt. Λ and π. Moreover, $I[\Lambda, \pi] \models C \leftarrow c$, for every $C \leftarrow c \in \Phi$.*

7 Derivations

We introduce one more inference rule, Simp, a generic simplification rule. In that, form denotes a (set of) clause(s) or a context as an ordinary formula, i.e., $\mathrm{form}(C \leftarrow R \cdot c) = \forall(C \vee \neg R \vee c)$. and $\mathrm{form}(\Lambda) = \{\forall(L^{\$} \vee \neg\mu^{1} c) \mid L|c \in \Lambda\}$, where $L^{\$}$ is obtained from L by replacing every F-sorted variable by the (same) F-sorted constant $\$$.

$$\Lambda \cdot \Gamma \vdash \Phi, D \quad \Longrightarrow \quad \Lambda \cdot \Gamma \vdash \Phi, D' \qquad (\mathsf{Simp})$$

if (i) D is redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi, D'$, and (ii) $\mathrm{form}(\Lambda) \cup \mathrm{form}(\Phi \cup D) \models \mathrm{form}(D')$. The first condition is needed for completeness, the second for soundness. The Simp rule encompasses various forms of simplification of the literals in C based on rewriting and subsumption and strictly covers the non-theory case [4].

We turn now to derivations, the process of deriving (saturated) sequents. All inference rules introduced so far are available for that. We will use κ to denote an at most countably infinite ordinal. Let Ψ be a set of input clauses and Γ a satisfiable set of closed constraints. The state $(\neg x|\top) \cdot \Gamma \vdash \{C \leftarrow \emptyset \cdot c \mid C \leftarrow c \in \Psi\}$ is called the *initial state for Ψ and Γ*. A *derivation $\mathcal{D}$ from Ψ and Γ* is a sequence $S_0 \Longrightarrow S_1 \Longrightarrow S_2 \Longrightarrow \cdots$ of states such that S_0 is the initial state for Ψ and Γ, and for every $i < \kappa$ there is an inference $S_i \Longrightarrow S_{i+1}$. A *refutation of Ψ and Γ* is a finite derivation from Ψ and Γ that ends in a Fail state.

Every derivation can be seen to generate in a depth-first, left-to-right fashion a *search tree*, a possibly infinite, ordered binary tree over states with root S_0 that branches out binary on a state S_i as determined by Decide and Minsol rules (left) and $\mathsf{Backtrack}$ inferences (right) to it. By a *branch $\mathbf{B}$ (in a derivation $\mathcal{D}$)* we mean any branch in $\mathcal{D}$'s search tree. The states in $\mathbf{B}$ are re-indexed for convenience as $\mathbf{B} = (\Lambda_i \cdot \Gamma_i \vdash \Phi_i)_{i<\kappa}$. Every branch $\mathbf{B}$ determines a possibly infinite *limit sequent* $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ where $\Lambda_{\mathbf{B}} = \bigcup_{i<\kappa} \Lambda_i$, $\Gamma_{\mathbf{B}} = \bigcup_{i<\kappa} \Gamma_i$ and $\Phi_{\mathbf{B}} = \bigcup_{i<\kappa} \bigcap_{i \le j<\kappa} \Phi_j$. The elements of these sets are called *persistent*. For space reasons we do not display here inference rules to simplify contexts, and so the stated definitions for $\Lambda_{\mathbf{B}}$ and $\Gamma_{\mathbf{B}}$ enough. The following definition is adapted from [4].

Definition 7.1 (Exhausted branch) A branch in $\mathcal{D}$ is *exhausted* iff for all $i < \kappa$:

(i) every inference with an evolution rule with premise $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ and persistent selected clause is redundant wrt. $\Lambda_j \cdot \Gamma_j \vdash \Phi_j$, for some j with $i \leq j < \kappa$, and

(ii) $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ is not closable, with no persistent clause.

□

We say that a derivation is *fair* iff it is a refutation or has an exhausted branch.

Proposition 7.2 (Fair derivations) *Let Ψ be a set of input clauses and Γ a satisfiable set of closed constraints. Suppose a fair derivation $\mathcal{D}$ from Ψ and Γ that is not a refutation, with limit branch $\mathbf{B}$. Then the limit sequent $\Lambda_\mathbf{B} \cdot \Gamma_\mathbf{B} \vdash \Phi_\mathbf{B}$ is saturated.*

Proposition 7.2 and Theorem 6.6 combined provide a refutational completeness result wrt fair derivations. For that, however, both conditions in Theorem 6.6 have to be satisfied, but they are non-trivial: firstly, if T is not compact then $\Gamma_\mathbf{B}$ could be unsatisfiable, although every background context derived is satisfiable. Consequently, although fair derivations always result in saturated limit sequents, unless they are refutations, satisfiability of the limit background contexts cannot be guaranteed. In this case Theorem 6.6 cannot be used to conclude satisfiability of Ψ. This is a theoretically acceptable. In practice, one could constrain all parameters to finite domains to ignore this problem. Secondly, the limit foreground context Λ will in general not provide a *total* interpretation for the B-sorted foreground operators, the underlying problem is Π^1_1-hard.

Theorem 7.3 (Relative refutational soundness) *Let Ψ be a set of input clauses and Γ a satisfiable set of closed constraints. Suppose a refutation from Ψ and Γ that Fails with some Γ'. Then, $\Gamma' \supseteq \Gamma$, Γ' is satisfiable, and $\Gamma' \cup \Psi$ is not satisfiable.*

Suppose the conditions of Theorem 7.3 hold, and let $I[\pi]$ be such that $\pi \models \Gamma'$, as claimed. It follows $I[\pi] \not\models \Psi$. 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, say, $\Gamma' = \{a > 2, a = 5\}$. Notice that $\pi \models \Gamma'$ entails $\pi = \{a \mapsto 5\}$, and, obviously, $I[\pi] \not\models \Psi$. But of course $\Psi \cup \Gamma$ is satisfiable, take, e.g., $\pi = \{a \mapsto 3\}$. A usual soundness result can thus be not based on *single* refutations, and this is why we call the soundness result above "relative". To fix that, we work with *sequences* of refutations whose limit background contexts collectively cover the initially given Γ. In the example, the next derivation starts with (essentially) $\Gamma = \{a > 2, a \neq 5\}$, which leads to a derivation that provides the expected model. See [5] for a general proof procedure based on this idea, which results in a sound calculus in the usual sense.

8 Deciding BS_T clause logic

The calculus provides a decision procedure for the satisfiability problem of (finite) sets of *BS_T-clauses*, that is, formulas of the form $\bar{\forall}\ C$, where C is an ordinary Σ-clause all whose functional subterms are ground. Notice that BS_T clause logic captures the

Bernays-Schönfinkel fragment of first-order logic, which is decided by all instance-based method. More interestingly, perhaps, it also lifts the functionality of current SMT-solvers as decision procedures for the important combination of LIA with the theory of free function symbols from the quantifier-free case to the said first-order fragment.

This is done as follows. Let Δ be a given set of BS_T clauses. Assume, for simplicity, Δ contains no parameters. That is, all B-sorted constants in Δ are declared as foreground operators. Instead of working with Δ directly, Δ is preprocessed so that every n-ary B-sorted foreground operators f (in Δ) occurs only in positive unit clauses of the form $f(d_1, \ldots, d_n) \approx a_{n+1}$, where the d_i's are ground terms whose only B-sorted subterms are parameters and a_{n+1} is a parameter. This form can always be achieved by extracting subterms and naming them with fresh parameters, in a satisfiability preserving way. For example, suppose Ψ contains the clause $f(g+1)+1 \not\approx h \vee P(x, i(j), h, f(1)) \vee g - 1 < 0$, where f, g and h are B-sorted foreground opertors. A preprocessed form then consists of $g \approx a_5$, $f(a_1) \approx a_6$, $f(a_8) \approx a_3$, $h \approx a_7$, $h \approx a_2$, $a_5 + 1 \approx a_1$, $a_6 + 1 \not\approx a_7 \vee P(x, i(j), a_2, a_3) \vee a_4 < 0$, $a_8 \approx 1$, $a_5 - 1 \approx a_4$, where the a_i's are parameters. Now, the calculus then works — without any change — with the preprocessed version turned into a set Ψ of normalized input clauses.

It is easy to see that every clause in Ψ is the normalized version of a clause of the form $f(d_1, \ldots, d_n) \approx a_{n+1}$ above or of a clause of the form $L_1 \vee \cdots \vee L_k \leftarrow c$ where all functional subterms in L_i are all ground and whose B-sorted subterms are all parameters. With Proposition 7.2 and Theorems 6.6 and 7.3 in place, it only remains to argue that (i) every fair derivation from Ψ and their background contexts in the sequence of derivations (as explained at the end of Section 7) terminates, (ii) that there are only finitely many such derivations, and (iii) if one of them is not a refutation then Ψ is satisfiable. Regarding (i), all Para inferences replace a functional term by a parameter, which results in a strictly smaller clause. Thus, only finitely many Para steps can be carried out in a derivation, and so only finitely many constrained empty clauses can be derived. The other inference rules merely analyze the boolean structure of these clauses (in a search tree), and there are only finitely many ways of doing that. Regarding (ii), new constraints in background contexts can be built only as equations between parameters resulting from paramodulation. In terms of the example above, the two equations $f(a_1) \approx a_6$ and $f(a_8) \approx a_3$ may end up as new constraints $a_1 = a_8$ and $a_6 = a_3$ in a derived background context. It is clear that only finitely many such equations exist. Because all other constraints in all constraint clauses are ground and fixed a priori, there are only finitely many non-equivalent boolean combinations over these constraints. Analyzing these combinations is essentially what happens in the mentioned sequence of derivations (see again [5]). Regarding (iii), the key point is that for every B-sorted foreground operator f a unit clause $f(d_1, \ldots, d_n) \approx a_{n+1}$ has been specified. These clauses all lead to rewrite rules in the induced interpretation $I[\Lambda, \pi]$ of the (finite) limit sequent $\Lambda \cdot \Gamma \vdash \Phi$, for any $\pi \in \mathrm{mods}(\Gamma)$, so that $\Lambda \cdot \Gamma \vdash \Phi$ is sufficiently complete. Theorem 6.6 then provides the model.

A Proofs

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

Lemma A.1 *Let l and r be pure terms. Then,*

(i) if $l \to r \in \mathcal{R}_{\Lambda,\pi}$ then Λ produces $l \approx r$, and

(ii) if $l \to r \notin \mathcal{R}_{\Lambda,\pi}$, $l \to r$ is suitable, and l and r are irreducible wrt $\mathcal{R}_{\Lambda,\pi}$ then Λ produces $l \not\approx r$ and Λ does not produce $l \approx r$.

Proof. The statement (i) follows immediately from the definition of $\mathcal{R}_{\Lambda,\pi}$.

Concerning (ii), suppose that l and r are irreducible wrt. $\mathcal{R}_{\Lambda,\pi}$. If Λ produces $l \approx r$ we distinguish two cases. If $\mathcal{R}_{\Lambda,\pi}$ generates $l \to r$ then l is reducible by $l \to r \in \mathcal{R}_{\Lambda,\pi}$. If $\mathcal{R}_{\Lambda,\pi}$ does not generate $l \to r$ then, by definition of $\mathcal{R}_{\Lambda,\pi}$, l or r must be reducible wrt. $(\mathcal{R}_{\Lambda,\pi})_{l \approx r}$, hence reducible wrt. $\mathcal{R}_{\Lambda,\pi}$. Both cases thus contradict the assumption that l and r are irreducible wrt. $\mathcal{R}_{\Lambda,\pi}$. 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.[6] Thus, with Λ not producing $l \approx r$ we can conclude that Λ produces $l \not\approx r$. □

By combining Definition 6.5 and Lemma A.1, conclude that, for relevant closures $(C \leftarrow R \cdot c, \gamma)$, Λ produces every (dis)-equation in $R\gamma$ and Λ does not produce $l \approx r$ for every $l \not\approx r \in R\gamma$. The relevance of this result is that relevant closures $(C \leftarrow R \cdot c, \gamma)$ that are falsified in an interpretation $I[\Lambda, \pi]$ can always be identified as those whose component $R\gamma$ is π-produced in the way stated.

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 6.2.)

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

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

(ii-a) in case of Para, *where σ is the mgu of the given inference, $l \approx r|d$ π-produces $(l \approx r|d)\sigma$ in Λ, $l \approx r|d$ π-produces $(l \approx r|d)\gamma$ in Λ, and $(l \approx r)\gamma$ generates the rule $(l \to r)\gamma$ in $\mathcal{R}_{\Lambda,\pi}$, and*

[6] Given Λ and L, thanks to the strict ordering $\dot{<}^{\pi}_{\mu}$, every sequence of context literals $(K)_{i \in I}$ such that $K_{i+1} \dot{<}^{\pi}_{\mu} K_i$ and K_i π-produces L in Λ (modulo sign) must be finite. This is an important detail to ensure that every context π-produces one of L or $\overline{L}$.

(ii-b) in case of Neg-Res *with selected literal* $s \approx t$*, used context literal* $\neg A|d$ *and mgu* σ*,* $\neg A$ *π-produces* $(s \not\approx t|d)\sigma$ *in* Λ*,* $\neg A$ *π-produces* $(s \not\approx t|d)\gamma$ *in* Λ*, and* $s\gamma$ *and* $t\gamma$ *are irreducible wrt.* $R_{\Lambda,\pi}$*.*

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

Proof. For convenience we abbreviate $R := \mathcal{R}_{\Lambda,\pi}$ below.

With (i), by relevancy, if $l \approx r \in R\gamma$ then $l \to r \in R$, and if $l \not\approx r \in R\gamma$ then l and r are irreducible wrt. R. Moreover, $(\Lambda, \pi) \models (R \cdot c, \gamma)$ means that $R\gamma$ is non-trivial, and for every $l \approx r \in R$, if $l\gamma \succ r\gamma$ then l is not a variable, and for every $K \in R$, Λ π-produces $K|c$ and $K\gamma|c\gamma$ by the same literal.

We have to show

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

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

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

(4) if $l \not\approx r \in R'\gamma$ 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 $K' \in R\sigma$ then Λ produces $K'|c'$ and $K'\gamma|c'\gamma$ by the same literals.

Proof: Assume $K' \in R\sigma$ and $c' = (c \wedge d)\sigma$, where d is absent in case of Ref. From (ii-a) and (ii-b) and by productivity it follows $\pi \models d\gamma$. Let $K \in R$ such that $K\sigma = K'$. We already know that some $L \in \Lambda$ π-produces $K|c$ in Λ and L π-produces $K\gamma|c\gamma$ in Λ. With $\pi \models d\gamma$ it follows trivially that L π-produces $K\gamma|(c \wedge d)\gamma$ in Λ. If L didn't π-produce $K\sigma|(c \wedge d)\sigma$ in Λ then there would be a $L' \in \Lambda$ with $\overline{L'} \dot{<}^{\pi}_{\mu} L$ and $\overline{L'} \gtrsim^{\pi} K\sigma|(c \wedge d)\sigma$. With $\gamma = \sigma\delta$ and by transitivity of $\gtrsim$ and stability of $\gtrsim$ under application of substitutions, we would get $\overline{L'} \gtrsim^{\pi} K\gamma|(c \wedge d)\gamma$, and so L would not π-produce $K\gamma|(c \wedge d)\gamma$ either.

Because L π-produces $K\gamma|(c \wedge d)\gamma$ in Λ, with $K\sigma = K'$, $(c \wedge d)\sigma = c$ and $\gamma = \sigma\gamma$ conclude that L π-produces $K'\gamma|c'\gamma$ in Λ, too.

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

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

(iii) if $l \not\approx r \in R\sigma\gamma$ then l and r are irreducible wrt. R.

Proof: we already know that if $l \not\approx r \in R\gamma$ 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 $C \leftarrow R \cdot c = L[s] \vee C'' \leftarrow R \cdot c$ and the conclusion $C' \leftarrow R' \cdot c' = (L[r] \vee C'' \leftarrow R \cup \{l \approx r\} \cdot c \wedge d)\sigma$. 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 $l \approx r|d$ π-produces $(l \approx r|d)\sigma$ in Λ and that $l \approx r|d$ π-produces $(l \approx r|d)\gamma$ in Λ. The proof that $l \approx r|d$ π-produces $(l \approx r|(c \wedge d))\sigma$ in Λ and that $l \approx r|d$ π-produces $(l \approx r|(c \wedge d))\gamma$ in Λ, which proves (2), is similar to the proof of (i) above and is omitted.

Regarding (3), recall we are given that $(l \approx r)\gamma$ generates $(l \to r)\gamma$ in R, which entails $(l \to r)\gamma = (l \to r)\sigma\gamma \in R$.

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

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

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

□

In words, $\Phi^{\Lambda,\pi}_{\mathcal{D}}$ is the set of relevant closures wrt. Λ and π of all clauses from Φ that are all smaller wrt. the closure ordering $\succ$ than $\mathcal{D}$.

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

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

If $(\Lambda, \pi) \not\models (R \cdot c, \gamma)$ then the conclusion follows trivially. Hence assume $(\Lambda, \pi) \models (R \cdot c, \gamma)$ from now on. It follows that case (a) in Definition 6.2 does not apply. Hence, case (b) in Definition 6.2 gives us closures $(C_i \leftarrow R_i \cdot c_i, \gamma_i)$ of clauses in Φ that satisfy properties (i) – (iv) in Definition 6.2.

From property (i) in Definition 6.2 it follows that each $(C_i \leftarrow R_i \cdot c_i, \gamma_i)$ is a relevant closure wrt. Λ and π. By condition (iii) in Definition 6.2, $(C_i \leftarrow R_i \cdot c_i, \gamma_i)$ is smaller

wrt. $\succ$ than $\mathcal{D}$. More formally, thus, $(C_i \leftarrow R_i \cdot c_i,\, \gamma_i) \in \Phi_{\mathcal{D}}^{\Lambda,\pi}$, and with (iii) conclude $I[\Lambda,\pi] \models (C_i \leftarrow R_i \cdot c_i,\, \gamma_i)$. From $(\Lambda,\pi) \models (R \cdot c, \gamma)$ and again with property (i) it follows $(\Lambda,\pi) \models (R_i \cdot c_i, \gamma_i)$, which entails $I[\Lambda,\pi] \models (C_i \leftarrow c_i)\gamma_i$ by Definition 4.2.

Again with $(\Lambda,\pi) \models (R \cdot c, \gamma)$ conclude $\pi \models c\gamma$ (by Definition 4.1). From property (ii) in Definition 6.2 it follows $\pi \models c_i\gamma_i$. Together with$I[\Lambda,\pi] \models (C_i \leftarrow c_i)\gamma_i$ it follows $I[\Lambda,\pi] \models C_i\gamma_i$. By property (iv) of redundancy, $\{C_1\gamma_1, \ldots, C_n\gamma_n\} \models C\gamma$, and so $I[\Lambda,\pi] \models C\gamma$ and trivially $I[\Lambda,\pi] \models (C \leftarrow c)\gamma$. With Definition 4.2 conclude $I[\Lambda,\pi] \models (C \leftarrow R \cdot c, \gamma)$, as desired. □

Proposition A.5 *Suppose $\pi \in \mathrm{mods}(\Gamma)$. Let $\Lambda \cdot \Gamma \vdash \Phi$ be a sequent and $(C \leftarrow R \cdot c, \gamma)$ a 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 π, and (iii) $I[\Lambda,\pi] \models \Phi^{\Lambda,\pi}_{(C \leftarrow R\cdot c,\gamma)}$ then $I[\Lambda,\pi] \models (C \leftarrow R \cdot c, \gamma)$.*

Proposition A.5 establishes a relationship between redundant clauses and satisfaction by $I[\Lambda,\pi]$. 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 π that are smaller wrt. $\succ$ than a hypothetically falsified closure $(C \leftarrow R \cdot c, \gamma)$ are all satisfied by $I[\Lambda,\pi]$. Proposition A.5 then allows to conclude that $(C \leftarrow R \cdot c, \gamma)$ is satisfied by $I[\Lambda,\pi]$, and hence cannot be that hypothetically falsified clause.

Proof. Immediate from Lemma A.4 by setting $\mathcal{D} = (C \leftarrow R \cdot c, \gamma)$, and using Definition 6.2. □

Theorem 6.6 (Static completeness) *Let $\Lambda \cdot \Gamma \vdash \Phi$ be a saturated sequent that is not closable and π a valuation. If $\pi \models \Gamma$ and $\Lambda \cdot \Gamma \vdash \Phi$ is sufficiently complete, then every total extension of $I[\Lambda,\pi]$ satisfies all Herbrand closures of all clauses in Φ that are relevant wrt. Λ and π. Moreover, $I[\Lambda,\pi] \models C \leftarrow c$, for every $C \leftarrow c \in \Phi$.*

Proof. Suppose $\pi \models \Gamma$, i.e., $\pi \in \mathrm{mods}(\Gamma)$ and that $I[\Lambda,\pi]$ has already been extended to be total on the B-sorted foreground operators. For the proof of the first statement we show the following property (P):

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

for every relevant closure $(C \leftarrow R \cdot c, \gamma)$ wrt. Λ and π 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 $(\Lambda, \pi, I[\Lambda,\pi]) \models (C \leftarrow \emptyset \cdot c,\, \gamma)$ if and only if $I[\Lambda,\pi] \models (C \leftarrow c)\gamma$.

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 π and that does not satisfy (P) must satisfy the following properties:

(i) for every $l \approx r \in R\gamma$, if $l \succ r$ then $l \to r \in \mathcal{R}_{\Lambda,\pi}$, and for every $l \not\approx r \in R\gamma$, l and r are irreducible wrt. $\mathcal{R}_{\Lambda,\pi}$, by Definition 6.5.

(ii) $(\Lambda,\pi) \models (R \cdot c, \gamma)$, and

(iii) $I[\Lambda, \pi] \not\models (C \leftarrow c)\gamma$, from $(C \leftarrow R \cdot c, \gamma)$ not satisfying (P) by Definition 4.2.

(iv) $\pi \models c\gamma$, from (ii), and

(v) $\mathcal{R}_{\Lambda,\pi} \not\models C\gamma$, from (iii), (iv) , by definition of induced interpretation and sufficient completeness (by sufficient completeness, $I[\Lambda, \pi] \models C\gamma$ iff $\mathcal{R}_{\Lambda,\pi} \models C\gamma$).

Among all counterexamples, by well-foundedness of the ordering $\succ$ on Herband closures, there is a minimal counterexample (minimal wrt. $\succ$). 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 π that is smaller wrt. $\succ$ than $(C \leftarrow R \cdot c, \gamma)$ satisfies (P). More formally, $I[\Lambda, \pi] \models \Phi^{\Lambda,\pi}_{(C \leftarrow R \cdot c, \gamma)}$.

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 $\mathcal{D} = (C \leftarrow R \cdot c, \gamma)$ there, (P) follows immediately, contradicting our assumption. Hence, $(C \leftarrow R \cdot c, \gamma)$ cannot be redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$.

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

First we show that every term $t \in (\mathrm{var}(C) \cap \mathrm{var}(R))\gamma$ is irreducible wrt. $\mathcal{R}_{\Lambda,\pi}$. For, if t is reducible wrt. $\mathcal{R}_{\Lambda,\pi}$ and occurs in a disequation $l \not\approx r \in R\gamma$ then we get a contradiction to (i). If t occurs in a positive literal $l \approx r \in R\gamma$ we conclude as follows: from (ii) it follows $l \neq r$ and hence, w.l.o.g, $l \succ r$. As $l \approx r \in R\gamma$ there is a $l' \approx r' \in R$ such that $(l' \approx r')\gamma = l \approx r$. By Definition 4.1-(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 $\mathcal{R}_{\Lambda,\pi}$, and hence $l \to r$ cannot be generated in $\mathcal{R}_{\Lambda,\pi}$, again contradicting (i).

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

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

With Lemma A.1-(i) it follows that Λ produces $l \approx r$, more precisely, the normalized version of the context literal $l \approx r|\top$. For later use let $l' \approx r'|d' \in \Lambda$ be a fresh variant

that produces that normalized version, which can be written as $(l' \approx r' | d')\gamma$. Notice that $(l' \approx r')\gamma = l \approx r$.

The conclusions so far give that Para is applicable with selected context equation $(l' \approx r' | d')\gamma$, selected clause $s\gamma[l'\gamma]_p \not\approx t\gamma \vee D\gamma \leftarrow R\gamma \cdot c\gamma$ and derived clause $s\gamma[r'\gamma]_p \not\approx t\gamma \vee D\gamma \leftarrow R\gamma \cup \{(l' \approx r')\gamma\} \cdot c\gamma \wedge d\gamma$. The next step is to show that this inference is a ground instance via γ of a Para inference with selected context equation $l' \approx r' | d'$, selected clause $C \leftarrow R \cdot c = s[u]_p \not\approx t \vee D \leftarrow R \cdot c$ and derived clause $(s[r']_p \not\approx t \vee D \leftarrow R \cup \{l' \approx r'\} \cdot c \wedge d)\sigma$, 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 $(l' \to r')\gamma = l \to r$, 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. Moreover, by suitability, if $l \to r$ contains a B-sorted foreground operator, it can only be at the top position of l. It follows that σ is simple. As all rules in $\mathcal{R}_{\Lambda,\pi}$ are ordered wrt. $\succ$, from $l \to r \in \mathcal{R}_{\Lambda,\pi}$ 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 $\mathcal{C} := ((s[r']_p \not\approx t \vee D \leftarrow R \cup \{l' \approx r'\} \cdot c \wedge d)\sigma, \gamma)$. 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 π, hence, by Lemma A.4, $I[\Lambda, \pi] \models \mathcal{C}$. By Definition 4.2, this means $(\Lambda, \pi) \not\models (R \cup \{l' \approx r'\} \cdot c \wedge d)\sigma, \gamma)$ or $I[\Lambda, \pi] \models (s[r']_p \not\approx t \vee D \leftarrow c \wedge d)\sigma\gamma$. However, Lemma A.2 gives us additionally $(\Lambda, \pi) \models ((R \cup \{l' \approx r'\} \cdot c \wedge d)\sigma, \gamma)$, and so the former case is impossible. By Definition 4.1 it follows $\pi \models (c \wedge d)\sigma\gamma$. With that, from $I[\Lambda, \pi] \models (s[r']_p \not\approx t \vee D \leftarrow c \wedge d)\sigma\gamma$ and by definition of induced interpretation it follows $\mathcal{R}_{\Lambda,\pi} \models (s[r']_p \not\approx t \vee D)\sigma\gamma$. With $(l' \to r')\gamma \in \mathcal{R}_{\Lambda,\pi}$ by congruence and $\sigma\gamma = \gamma$ it follows $\mathcal{R}_{\Lambda,\pi} \models (s \not\approx t \vee D)\gamma$. Using $C = s \not\approx t \vee D$ this is a plain contradiction to (v) above.

(3.2) $s\gamma = t\gamma$.
If $s\gamma = t\gamma$ then 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 $(D \leftarrow R \cdot c)\sigma$, unless σ is not simple.

Let us consider the case that σ is not simple. This means that some subterm of, say, s starts with a B-sorted foreground operator that is unified with a variable at the same position in t. But then, by sufficient completeness, $C\gamma$ and hence in particular $s\gamma \not\approx t\gamma$ is reducible wrt. $\mathcal{R}_{\Lambda,\pi}$ to an equivalent disequation. In this case the same argumentation as in case (3.1) applies.

Hence assume that σ is simple. 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 $\mathcal{C} := ((D \leftarrow R \cdot c)\sigma, \gamma)$. 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 π, hence, by Lemma A.4, $I[\Lambda, \pi] \models \mathcal{C}$. By Definition 4.2, this means $(\Lambda, \pi) \not\models (R \cdot c)\sigma, \gamma)$ or $I[\Lambda, \pi] \models (D \leftarrow c)\sigma\gamma$. However, Lemma A.2 gives us additionally $(\Lambda, \pi) \models ((R \cdot c)\sigma, \gamma)$, and so the former case is impossible. But then, from $I[\Lambda, \pi] \models (D \leftarrow c)\sigma\gamma$ and $\sigma\gamma = \gamma$ by definition of induced interpretation it follows $\pi \not\models c\gamma$ or $\mathcal{R}_{\Lambda,\pi} \models D\gamma$. However, by (iv) the first case is impossible, and in the second case, trivially, $\mathcal{R}_{\Lambda,\pi} \models C\gamma$, 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 (ii), $s\gamma = t\gamma$ is impossible. Hence assume $s\gamma \neq t\gamma$ in the following. We distinguish two further cases.

(4.1) $s\gamma$ *or* $t\gamma$ *is reducible wrt.* $\mathcal{R}_{\Lambda,\pi}$.
If $s\gamma$ or $t\gamma$ is reducible wrt. $\mathcal{R}_{\Lambda,\pi}$ then there are ground terms l and r such that there is a rule $l \to r \in \mathcal{R}_{\Lambda,\pi}$ that rewrites $s\gamma$ or $t\gamma$. With the same argumentation as in case (3.1) one shows that a smaller, yet congruent counterexample would exists due to a Para inference, which leads to the same contradiction as in case (3.1).

(4.2) $s\gamma$ *and* $t\gamma$ *are irreducible wrt.* $\mathcal{R}_{\Lambda,\pi}$.
This case is meant to say that case (4.1) does not apply, for no equation $s \approx t \in C$. Thus, for every $s \approx t \in C$, $s\gamma$ and $t\gamma$ are irreducible wrt. $\mathcal{R}_{\Lambda,\pi}$. Assume, w.l.o.g., $s\gamma \succ t\gamma$. With Lemma A.1-(ii) then conclude that some literal $\neg A|d \in \Lambda$ produces $(s \not\approx t)\gamma$ in Λ. This indicates that a ground Neg-Res inference exists. It is routine by now to check that this ground Neg-Res inference is a ground instance via γ of a Neg-Res inference. The rest of the proof uses the same arguments as in case (3.1) and is omitted.

(5) $C = \square$.
Suppose $C = \square$. First we are going to show that Decide-1 or Decide-2 is applicable to $\Lambda \cdot \Gamma \vdash \Phi$ with selected clause $\square \leftarrow R \cdot c$.

With property (ii) above, there is a $K_i|d_i \in \Lambda$ that π-produces $L_i|c$ and $L_i\gamma|c\gamma$ in Λ, for every $L_i \in R$. Without loss of generality assume the literals $K_i|d_i$ are chosen minimal wrt the strict ordering $\dot{<}^{\pi}_{\mu}$ among all context literals with that property.

In the following we take the substitution γ also as a solution $\vec{x}\gamma$ of a constraint with free variables $\vec{x}$.

Consider the case that for every i, γ is a π-solution of $\mu^1 d_i$ or a minimal π-solution of d_i, and that $L_i = K_i$. Then, Minsol inferences exist to eliminate all latter cases by comitting to these minimal solutions, as certain k-th least solutions of the relevant constraints d_i. The context literal to be added by Minsol cannot be π-contradictory with Λ, because, if so, Λ would not π-produce the corresponding context literal $L_i\gamma|c\gamma$. And the complement of the added context literal cannot be π-contradictory with Λ, because, if so, Λ would already contain a literal equivalent to $L_i|\mu^1 c$. By saturation, all these inference must have been carried out, or others that have the same effect. But then Λ is closable, a plain contradiction.

Hence for some i, the above case does not apply, and we consider the complementary case.

In the first subcase $L_i \neq K_i$. That means, L_i is a proper instance of K_i. The literal $\overline{L_i}|c$ cannot be π-contradictory with Λ because then Λ contains a literal that is $\sim^\pi$-equivalent to $L_i|c$ and that is smaller wrt. $\dot{<}^p_\mu$ i than $K_i|d_i$ and that π-produces $L_i\gamma|c\gamma$, contradicting the choice of the $K_i|d_i$ above. This shows that a Decide-1 inference with $\overline{L_i}|c$ exists.

In the second subcase $L_i = K_i$. It follows γ is not a minimal π-solution of d_i. But then π satisfies not only $c\gamma$ but also $(c \wedge \neg\mu d_i)\gamma$. Recall that $K_i|d_i$ π-produces $L_i|c$ and $L_i\gamma|c\gamma$. It follows that $K_i|d_i$ π-produces (trivially) $L_i\gamma|(c \wedge \neg\mu d_i)\gamma$ and also that $K_i|d_i$ π-produces $L_i\gamma|(c \wedge \neg\mu d_i)$. Furthermore, $\overline{L}|c \wedge \neg\mu d_i$ is not π-contradictory with Λ because then Λ contains a literal that is $\sim^\pi$-equivalent to $L_i|c \wedge \neg\mu d_i$ and that is smaller wrt. $\dot{<}^p_\mu$ i than $K_i|d_i$ and that π-produces $L_i\gamma|c\gamma$, again contradicting the choice of the $K_i|d_i$ above. This shows that a Decide-2 inference with $\overline{L_i}|c \wedge \neg\mu d_i$ exists.

By saturation, Λ does not produce $L_i|c$ (first subcase) and Λ does not produce $L_i|c \wedge \neg\mu d_i$ (second subcase), in contradiction to assumptions above that $K_i|d_i$ π-produces these literals in Λ.

This covers all cases. Each case led to a contradiction, hence the assumption of the existence of a counterexample is invalid, and the theorem is proven. □

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 context 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. 6.2 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 context literal is added to Λ one needs to prove that, in terms of Def. 6.2, $(C \leftarrow R \cdot c, \gamma)$ remains redundant wrt. $\Lambda \cdot \Gamma \vdash \Phi$ and $\mathcal{D}$. 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 6.2 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 $\mathcal{D}$. 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 that satisfies the conditions of Definition 6.2. 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 6.2 applies to $(C_1 \leftarrow R_1 \cdot c_1, \gamma_1)$ then from Definition 6.2-(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 6.2 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 6.2 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* Neg-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 context 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 inference is redundant wrt.* $\Lambda' \cdot \Gamma' \vdash \Phi'$.

Proof. The non-trivial case is, in terms of Definition 6.3, 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* Neg-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. □

The following proposition is instrumental for linking fair derivations with the static completeness result, Theorem 6.6.

Proposition 7.2 (Fair derivations) *Let* Ψ *be a set of input clauses and* Γ *a satisfiable set of closed constraints. Suppose a fair derivation* $\mathcal{D}$ *from* Ψ *and* Γ *that is not a refutation, with limit branch* $\mathbf{B}$. *Then the limit sequent* $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ *is saturated.*

Proof. We have to show that every inference with an evolution inference rule and premise $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is redundant wrt. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$, and that $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is not closable with D, for no $D \in \Phi_{\mathbf{B}}$. For the first part we assume such an inference and carry out a case analysis wrt. the inference rule applied.

If the inference rule is Decide-1 or Decide-2 then let $\square \leftarrow R \cdot c$ be the selected clause. There are only finitely many literals, modulo equivalence and modulo sign, that are larger than the literal to be added to Λ wrt. the strict order $\dot{<}^{\pi}_{\mu}$ for a fixed π. This entails that from some timepoint on the addition of context literals can no longer influence whether such inferences are redundant (or not). In consequence, making every inference (with persistent clauses) redundant eventually ensures that it will be redundant wrt.

$\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. Whether mods($\Gamma$) is finite or not is not important for that; it suffices to achieve redundancy for each $\pi \in$ mods(Γ) individually.

A similar argumentation applies in case of Minsol.

If the inference rule is Neg-Res, Ref or Para then by Definition 7.1 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. $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$.

By Definition 7.1 $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ is not closable, for no $i < \kappa$, with a persistent clause. But then $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is not closable either. (Because if so, for a large enough i, $\Lambda_i \cdot \Gamma_i \vdash \Phi_i$ would be closable with some persistent clause.) □

Theorem 7.3 (Relative refutational soundness) *Let Ψ be a set of input clauses and Γ a satisfiable set of closed constraints. Suppose a refutation from Ψ and Γ that Fails with some Γ'. Then, $\Gamma' \supseteq \Gamma$, Γ' is satisfiable, and $\Gamma' \cup \Psi$ is not satisfiable.*

Proof. (Sketch) Theorem 7.3 can be proven as follows. Let $\mathcal{D}$ be the given refutation, $\mathbf{B}$ the rightmost branch in its search tree and $\Lambda_{\mathbf{B}} \cdot \Gamma_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ the limit sequent of $\mathbf{B}$, which is the same as the leaf of $\mathbf{B}$ (as refutations are finite), and so $\Gamma' = \Gamma_{\mathbf{B}}$. The conclusions $\Gamma' \supseteq \Gamma$ and that Γ' is satisfiable in Theorem 7.3 follow easily by construction. Now chose any interpretation I such that $I \models \Gamma'$ arbitrarily. For the last conclusion it suffices, in its core, to show (i) that *$\mathcal{D}$ preserves I-satisfaction*: for every non-leaf node $\Lambda \cdot \Gamma \vdash \Phi$ in $\mathcal{D}$'s search tree, if $I \models \text{form}(\Lambda) \cup \text{form}(\Phi)$ then $I \models \text{form}(\Lambda') \cup \text{form}(\Phi')$ for some child node $\Lambda' \cdot \Gamma' \vdash \Phi'$ of $\Lambda \cdot \Gamma \vdash \Phi$ (this follows from the design of the inference rules). Furthemore it needs to be shown that, (ii), I does not satisfy any leaf node $\Lambda \cdot \Gamma \vdash \Phi$, which is closed by Backtrack/Fail inferences (in each of them, the closing constraint provides a constrained clause that is false in I as an ordinary formula). □

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

[3] 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.

[4] P. Baumgartner, B. Pelzer, and C. Tinelli. Model evolution with equality – revised and implemented. *Journal of Symbolic Computation*, 2011. Available Online. In Press.

[5] P. Baumgartner and C. Tinelli. Model evolution with equality modulo built-in theories. In N. Bjoerner and V. Sofronie-Stokkermans, editors, *CADE-23 – The 23nd International Conference on Automated Deduction*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 85–100. Springer, 2011.

[6] 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.

[7] 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.

[8] Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2009.

[9] 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.

[10] 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.

[11] 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.