Journal of Automated Reasoning (2023) 67:22
https://doi.org/10.1007/s10817-023-09673-3

SCL(EQ): SCL for First-Order Logic with Equality

Hendrik Leidinger[1,2] · **Christoph Weidenbach**[1]

Received: 8 February 2023 / Accepted: 6 June 2023 / Published online: 30 June 2023

Abstract
We propose a new calculus SCL(EQ) for first-order logic with equality that only learns non-redundant clauses. Following the idea of CDCL (Conflict Driven Clause Learning) and SCL (Clause Learning from Simple Models) a ground literal model assumption is used to guide inferences that are then guaranteed to be non-redundant. Redundancy is defined with respect to a dynamically changing ordering derived from the ground literal model assumption. We prove SCL(EQ) sound and complete and provide examples where our calculus improves on superposition.

Keywords First-order logic with equality · Term rewriting · Model-based reasoning

1 Introduction

There has been extensive research on sound and complete calculi for first-order logic with equality. The current prime calculus is superposition [2], where ordering restrictions guide paramodulation inferences and an abstract redundancy notion enables a number of clause simplification and deletion mechanisms, such as rewriting or subsumption. Still this "syntactic" form of superposition infers many redundant clauses. The completeness proof of superposition provides a "semantic" way of generating only non-redundant clauses, however, the underlying ground model assumption cannot be effectively computed in general [35]. It requires an ordered enumeration of infinitely many ground instances of the given clause set, in general. Our calculus overcomes this issue by providing an effective way of generating ground model assumptions that then guarantee non-redundant inferences on the original clauses with variables.

The underlying ordering is based on the order of ground literals in the model assumption, hence changes during a run of the calculus. It incorporates a standard rewrite ordering. For

✉ Hendrik Leidinger
hleiding@mpi-inf.mpg.de

Christoph Weidenbach
weidenbach@mpi-inf.mpg.de

[1] Max-Planck Institute for Informatics, Saarland Informatics Campus E1 4, Saarbrücken 66123, Germany

[2] Graduate School of Computer Science, Saarland Informatics Campus E1 3, Saarbrücken 66123, Germany

practical redundancy criteria this means that both rewriting and redundancy notions that are based on literal subset relations are permitted to dynamically simplify or eliminate clauses. Newly generated clauses are non-redundant, so redundancy tests are only needed backwards. Furthermore, the ordering is automatically generated by the structure of the clause set. Instead of a fixed ordering as done in the superposition case, the calculus finds and changes an ordering according to the currently easiest way to make progress, analogous to CDCL (Conflict Driven Clause Learning) [11, 12, 28, 33, 39].

Typical for CDCL and SCL (Clause Learning from Simple Models) [1, 15, 21] approaches to reasoning, the development of a model assumption is done by decisions and propagations. A decision guesses a ground literal to be true whereas a propagation concludes the truth of a ground literal through an otherwise false clause. While propagations in CDCL and propositional logic are restricted to the finite number of propositional variables, in first-order logic there can already be infinite propagation sequences [21], as there might exist an infinite set of ground instances. In order to overcome this issue, model assumptions in SCL(EQ) are at any point in time restricted to a finite number of ground literals, hence to a finite number of ground instances of the clause set at hand. Therefore, without increasing the number of considered ground literals, the calculus either finds a refutation or runs into a *stuck state* where the current model assumption satisfies the finite number of ground instances. In this case one can check whether the model assumption can be generalized to a model assumption of the overall clause set or the information of the stuck state can be used to appropriately increase the number of considered ground literals and continue search for a refutation. SCL(EQ) does not require exhaustive propagation, in general, it just forbids the decision of the complement of a literal that could otherwise be propagated.

For an example of SCL(EQ) inferring clauses, consider the three first-order clauses

$$C_1 := h(x) \approx g(x) \vee c \approx d \qquad C_2 := f(x) \approx g(x) \vee a \approx b$$
$$C_3 := f(x) \not\approx h(x) \vee f(x) \not\approx g(x)$$

with a Knuth–Bendix Ordering (KBO), unique weight 1, and precedence $d \prec c \prec b \prec a \prec g \prec h \prec f$. A Superposition Left [2] inference between C_2 and C_3 results in

$$C_4' := h(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee a \approx b.$$

For SCL(EQ) we start by building a partial model assumption, called a *trail*, with two decisions

$$\Gamma := \left[h(a) \approx g(a)^{1:(h(x)\approx g(x) \vee h(x) \not\approx g(x)) \cdot \sigma}, \; f(a) \approx g(a)^{2:(f(x)\approx g(x) \vee f(x) \not\approx g(x)) \cdot \sigma} \right],$$

where $\sigma := \{x \mapsto a\}$. Decisions and propagations are always ground instances of literals from the first-order clauses, and are annotated with a level and a justification clause, in case of a decision a tautology. Now with respect to Γ clause C_3 is false with grounding σ, and rule Conflict is applicable; see Sect. 3.1 for details on the inference rules. In general, clauses and justifications are considered variable disjoint, but for simplicity of the presentation of this example, we repeat variable names here as long as the same ground substitution is shared. The maximal literal in $C_3\sigma$ is $(f(x) \not\approx h(x))\sigma$ and a rewrite refutation using the ground equations from the trail results in the justification clause

$$(g(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee h(x) \not\approx g(x)) \cdot \sigma,$$

where for the refutation justification clauses and all otherwise inferred clauses we use the grounding σ for guidance, but operate on the clauses with variables. The respective ground clause is smaller than $(f(x) \not\approx h(x))\sigma$, false with respect to Γ and becomes our new conflict

clause by an application of our inference rule Explore-Refutation. It is simplified by our inference rules Equality-Resolution and Factorize, resulting in the finally learned clause

$$C_4 := h(x) \not\approx g(x) \vee f(x) \not\approx g(x),$$

which is then used to apply rule Backtrack to the trail. Further details on this example are available from the Appendix, Example 6. Observe that C_4 is strictly stronger than C_4' the clause inferred by superposition and that C_4 cannot be inferred by superposition. Thus SCL(EQ) can infer stronger clauses than superposition for this example.

1.1 Related Work

SCL(EQ) is based on ideas of SCL [1, 15, 17, 21] but for the first time includes a native treatment of first-order equality reasoning. Similar to [15] propagations need not to be exhaustively applied, the trail is built out of decisions and propagations of ground literals annotated by first-order clauses, SCL(EQ) only learns non-redundant clauses, but for the first time conflicts resulting out of a decision have to be considered, due to the nature of the equality relation.

There have been suggested several approaches to lift the idea of an inference guiding model assumption from propositional to full first-order logic [8, 13, 14, 21]. They do not provide a native treatment of equality, e.g., via paramodulation or rewriting.

Baumgartner et al. describe multiple calculi that handle equality by using unit superposition style inference rules and are based on either hyper tableaux [5] or DPLL [18, 19]. Hyper tableaux fix a major problem of the well-known free variable tableaux, namely the fact that free variables within the tableau are rigid, i.e., substitutions have to be applied to all occurrences of a free variable within the entire tableau. Hyper tableaux with equality [9] in turn integrates unit superposition style inference rules into the hyper tableau calculus.

Another approach that is related to ours is the model evolution calculus with equality ($\mathcal{ME}_{\mathcal{E}}$) by Baumgartner et al. [6, 10] which lifts the DPLL calculus to first-order logic with equality. Similar to our approach, $\mathcal{ME}_{\mathcal{E}}$ creates a candidate model until a clause instance contradicts this model or all instances are satisfied by the model. The candidate model results from a so-called context, which consists of a finite set of non-ground rewrite literals. Roughly speaking, a context literal specifies the truth value of all its ground instances unless a more specific literal specifies the complement. Initially the model satisfies the identity relation over the set of all ground terms. Literals within a context may be universal or parametric, where universal literals guarantee all its ground instances to be true. If a clause contradicts the current model, it is repaired by a non-deterministic split which adds a parametric literal to the current model. If the added literal does not share any variables in the contradictory clause it is added as a universal literal.

Another approach by Baumgartner and Waldmann [7] combined the superposition calculus with the Model Evolution calculus with equality. In this calculus the atoms of the clauses are labeled as "split atoms" or "superposition atoms". The superposition part of the calculus then generates a model for the superposition atoms while the model evolution part generates a model for the split atoms. Conversely, this means that if all atoms are labeled as "split atom", the calculus behaves similar to the model evolution calculus. If all atoms are labeled as "superposition atom", it behaves like the superposition calculus.

Both the hyper tableaux calculus with equality and the model evolution calculus with equality allow only unit superposition applications, while SCL(EQ) inferences are guided paramodulation inferences on clauses of arbitrary length. The model evolution calculus with equality was revised and implemented in 2011 [10] and compares its performance with that of

hyper tableaux. Model evolution performed significantly better, with more problems solved in all relevant TPTP [34] categories, than the implementation of the hyper tableaux calculus.

Plaisted et al. [31] present the Ordered Semantic Hyper-Linking (OSHL) calculus. OSHL is an instantiation based approach that repeatedly chooses ground instances of a non-ground input clause set such that the current model does not satisfy the current ground clause set. A further step repairs the current model such that it satisfies the ground clause set again. The algorithm terminates if the set of ground clauses contains the empty clause. OSHL supports rewriting and narrowing, but only with unit clauses. In order to handle non-unit clauses it makes use of other mechanisms such as Brand's Transformation [4].

Inst-Gen [25] is an instantiation based calculus, that creates ground instances of the input first-order formulas which are forwarded to a SAT solver. If a ground instance is unsatisfiable, then the first-order set is as well. If not then the calculus creates more instances. The Inst-Gen-EQ calculus [26] creates instances by extracting instantiations of unit superposition refutations of selected literals of the first-order clause set. The ground abstraction is then extended by the extracted clauses and an SMT solver then checks the satisfiability of the resulting set of equational and non-equational ground literals.

On ground equational clauses, the behavior of SCL(EQ) is similar to SMT (Satisfiability Modulo Theories) [30]. SCL(EQ) rigorously searches for implied equalities and does not explicitely consider the propositional abstraction that drives SMT. Therefore, SCL(EQ) only learns non-redundant clauses that is not guaranteed by standard SMT reasoning. On the other hand the level of laziness in reasoning that is offered by SMT is currently not supported by SCL(EQ). On equational clauses with variables, SCL(EQ) learns only non-redundant clauses with variables whereas SMT solely operates on ground instances.

This article is an extended version of our proceedings paper [27]. It includes more examples and an extended background and discussion section. In favor of a better structure we have moved all proofs to an Appendix. The rest of the paper is organized as follows. Section 2 provides basic formalisms underlying SCL(EQ). The rules of the calculus are presented in Sect. 3. Soundness and completeness results are provided in Sect. 4. We end with a discussion of obtained results and future work, Sect. 5. The main contribution of this paper is the SCL(EQ) calculus that only learns non-redundant clauses, permits subset based redundancy elimination and rewriting, and its soundness and completeness.

2 Preliminaries

We assume a standard first-order language with equality and signature $\Sigma = (\Omega, \emptyset)$ where the only predicate symbol is equality $\approx$. N denotes a set of clauses, C, D denote clauses, L, K, H denote equational literals, A, B denote equational atoms, t, s terms from $T(\Omega, \mathcal{X})$ for an infinite set of variables $\mathcal{X}$, f, g, h function symbols from Ω, a, b, c constants from Ω and x, y, z variables from $\mathcal{X}$. The function $comp$ denotes the complement of a literal. We write $s \not\approx t$ as a shortcut for $\neg(s \approx t)$. The literal $s \# t$ may denote both $s \approx t$ and $s \not\approx t$. The semantics of first-order logic and semantic entailment $\models$ is defined as usual.

By σ, τ, δ we denote substitutions, which are total mappings from variables to terms. Let σ be a substitution, then its finite domain is defined as $dom(\sigma) := \{x \mid x\sigma \neq x\}$ and its codomain is defined as $codom(\sigma) = \{t \mid x\sigma = t, x \in dom(\sigma)\}$. We extend their application to literals, clauses and sets of such objects in the usual way. A term, literal, clause or sets of these objects is ground if it does not contain any variable. A substitution σ is $ground$ if $codom(\sigma)$ is ground. A substitution σ is $grounding$ for a term t, literal L, clause C if $t\sigma$,

$L\sigma$, $C\sigma$ is ground, respectively. By $C \cdot \sigma$, $L \cdot \sigma$ we denote a closure consisting of a clause C, literal L and a grounding substitution σ, respectively. The function gnd computes the set of all ground instances of a literal, clause, or clause set. The function mgu denotes the most general unifier of terms, atoms, literals, respectively. We assume that mgus do not introduce fresh variables and that they are idempotent.

The set of positions $\text{pos}(L)$ of a literal (term $\text{pos}(t)$) is inductively defined as usual. The notion $L|_p$ denotes the subterm of a literal L ($t|_p$ for term t) at position $p \in \text{pos}(L)$ ($p \in \text{pos}(t)$). The replacement of a subterm of a literal L (term t) at position $p \in \text{pos}(L)$ ($p \in \text{pos}(t)$) by a term s is denoted by $L[s]_p$ ($t[s]_p$). For example, the term $f(a, g(x))$ has the positions $\{\epsilon, 1, 2, 21\}$, $f(a, g(x))|_{21} = x$ and $f(a, g(x))[b]_2$ denotes the term $f(a, b)$.

Let R be a set of rewrite rules $l \to r$, called a *term rewrite system* (TRS). The rewrite relation $\to_R \subseteq T(\Omega, \mathcal{X}) \times T(\Omega, \mathcal{X})$ is defined as usual by $s \to_R t$ if there exists $(l \to r) \in R$, $p \in \text{pos}(s)$, and a matcher σ, such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$. We write $s = t{\downarrow_R}$ if s is the normal form of t in the rewrite relation $\to_R$. We write $s \# t = (s' \# t'){\downarrow_R}$ if s is the normal form of s' and t is the normal form of t'. A rewrite relation is terminating if there is no infinite descending chain $t_0 \to t_1 \to \ldots$ and confluent if $t \,{}^*\!\!\leftarrow s \to^* t'$ implies $t \leftrightarrow^* t'$. A rewrite relation is convergent if it is terminating and confluent. A rewrite order is a irreflexive and transitive rewrite relation. A TRS R is terminating, confluent, convergent, if the rewrite relation $\to_R$ is terminating, confluent, convergent, respectively. A term t is called irreducible by a TRS R if no rule from R rewrites t. Otherwise it is called reducible. A literal, clause is irreducible if all of its terms are irreducible, and reducible otherwise. A substitution σ is called irreducible if any $t \in codom(\sigma)$ is irreducible, and reducible otherwise.

Let $\prec_T$ denote a well-founded rewrite ordering on terms which is total on ground terms and for all ground terms t there exist only finitely many ground terms $s \prec_T t$. We call $\prec_T$ a *desired* term ordering. We extend $\prec_T$ to equations by assigning the multiset $\{s, t\}$ to positive equations $s \approx t$ and $\{s, s, t, t\}$ to inequations $s \not\approx t$. Furthermore, we identify $\prec_T$ with its multiset extension comparing multisets of literals. For a (multi)set of terms $\{t_1, \ldots, t_n\}$ and a term t, we define $\{t_1, \ldots, t_n\} \prec_T t$ if $\{t_1, \ldots, t_n\} \prec_T \{t\}$. For a (multi)set of Literals $\{L_1, \ldots, L_n\}$ and a term t, we define $\{L_1, \ldots, L_n\} \prec_T t$ if $\{L_1, \ldots, L_n\} \prec_T \{\{t\}\}$. Given a ground term β then $\text{gnd}_{\prec_T \beta}$ computes the set of all ground instances of a literal, clause, or clause set where the groundings are smaller than β according to the ordering $\prec_T$. Given a set (sequence) of ground literals Γ let $\text{conv}(\Gamma)$ be a convergent rewrite system from the positive equations in Γ using $\prec_T$.

Let $\prec$ be a well-founded, total, strict ordering on ground literals, which is lifted to clauses and clause sets by its respective multiset extension. We overload $\prec$ for literals, clauses, clause sets if the meaning is clear from the context. The ordering is lifted to the non-ground case via instantiation: we define $C \prec D$ if for all grounding substitutions σ it holds $C\sigma \prec D\sigma$. Then we define $\preceq$ as the reflexive closure of $\prec$ and $N^{\preceq C} := \{D \mid D \in N \text{ and } D \preceq C\}$ and use the standard superposition style notion of redundancy [2].

Definition 1 *(Clause Redundancy)* A ground clause C is *redundant* with respect to a set N of ground clauses and an ordering $\prec$ if $N^{\preceq C} \models C$. A clause C is *redundant* with respect to a clause set N and an ordering $\prec$ if for all $C' \in gnd(C)$, C' is redundant with respect to $gnd(N)$.

The Superposition calculus [2] is defined by the following rules. Given a reduction order $\succ$ on terms which is extended to an ordering on literals and clauses in the usual way, the basic Superposition rule is as follows:

Superposition

$N \cup \{D \vee t \approx t', C \vee s[u] \# s'\} \Rightarrow_{\text{SUP}} N \cup \{D \vee t \approx t', C \vee s[u] \# s', (C \vee D \vee s[t'] \# s')\sigma\}$
where $\sigma = mgu(t, u)$ and u is not a variable.

Inferences are only allowed if the left premise is not greater than or equal to the right one, the last literal of each premise is greater than the remaining literals of the respective clause and the lefthand side of these literals is not smaller or equal to the righthand side. Two more rules are needed for completeness:

Equality Resolution

$N \cup \{C \vee s \not\approx s'\} \Rightarrow_{\text{SUP}} N \cup \{C\sigma\}$
where $\sigma = mgu(s, s')$

Equality Factoring

$N \cup \{C \vee s \not\approx t' \vee s \approx t\} \Rightarrow_{\text{SUP}} N \cup \{(C \vee t \not\approx t' \vee s \approx t')\sigma\}$
where $\sigma = mgu(s, s')$

3 The SCL(EQ) Calculus

We start the introduction of the calculus by defining the ingredients of an SCL(EQ) state.

Definition 2 *(Trail)* A *trail* $\Gamma := [L_1^{i_1:C_1 \cdot \sigma_1}, ..., L_n^{i_n:C_n \cdot \sigma_n}]$ is a consistent sequence of ground equations and inequations where L_j is annotated by a level i_j with $i_{j-1} \leq i_j$, and a closure $C_j \cdot \sigma_j$. We omit the annotations if they are not needed in a certain context. A ground literal L is true in Γ if $\Gamma \models L$. A ground literal L is false in Γ if $\Gamma \models comp(L)$. A ground literal L is undefined in Γ if $\Gamma \not\models L$ and $\Gamma \not\models comp(L)$. Otherwise it is defined. For each literal L_j in Γ it holds that L_j is undefined in $[L_1, ..., L_{j-1}]$ and irreducible by conv($\{L_1, ..., L_{j-1}\}$).

The above definition of truth and undefinedness is extended to clauses in the obvious way. The notions of true, false, undefined can be parameterized by a ground term β by saying that L is β-undefined in a trail Γ if $\beta \prec_T L$ or L is undefined. The notions of a β-true, β-false term are restrictions of the above notions to literals smaller β, respectively. All SCL(EQ) reasoning is layered with respect to a ground term β.

Definition 3 Let Γ be a trail and L a ground literal such that L is defined in Γ. By core(Γ; L) we denote a minimal subsequence $\Gamma' \subseteq \Gamma$ such that L is defined in Γ'. By cores(Γ; L) we denote the set of all cores.

Note that $core(\Gamma; L)$ is not necessarily unique. There can be multiple cores for a given trail Γ and ground literal L.

Definition 4 *(Trail Ordering)* Let $\Gamma := [L_1, ..., L_n]$ be a trail. The (partial) trail ordering $\prec_\Gamma$ is the sequence ordering given by Γ, i.e., ($L_i \prec_\Gamma L_j$ if $i < j$) for all $1 \leq i, j \leq n$.

Definition 5 *(Defining Core and Defining Literal)* For a trail Γ and a sequence of literals $\Delta \subseteq \Gamma$ we write $max_{\prec_\Gamma}(\Delta)$ for the largest literal in Δ according to the trail ordering $\prec_\Gamma$. Let Γ be a trail and L a ground literal such that L is defined in Γ. Let $\Delta \in cores(\Gamma; L)$ be a sequence of literals where $max_{\prec_\Gamma}(\Delta) \preceq_\Gamma max_{\prec_\Gamma}(\Lambda)$ for all $\Lambda \in cores(\Gamma; L)$, then $\max_\Gamma(L) := \max_{\prec_\Gamma}(\Delta)$ is called the *defining literal* and Δ is called a *defining core* for L in Γ. If $cores(\Gamma; L)$ contains only the empty core, then L has no *defining literal* and no *defining core*.

As an example, consider the trail $\Gamma{:=}[L_1, L_2, L_3]$ and literal L, such that $[L_1, L_2] \models L$ and $[L_2, L_3] \models L$. Then both $[L_1, L_2]$ and $[L_2, L_3]$ are cores, but only $[L_1, L_2]$ is a defining core and only L_2 is the defining literal. Note that there can be multiple defining cores but only one defining literal for any defined literal L. For example, consider a trail $\Gamma{:=}[f(a) \approx f(b)^{1:C_1\cdot\sigma_1}, a \approx b^{2:C_2\cdot\sigma_2}, b \approx c^{3:C_3\cdot\sigma_3}]$ with an ordering $\prec_T$ that orders the terms of the equations from left to right, and a literal $g(f(a)) \approx g(f(c))$. Then the defining cores are $\Delta_1{:=}[a \approx b, b \approx c]$ and $\Delta_2{:=}[f(a) \approx f(b), b \approx c]$. The defining literal, however, is in both cases $b \approx c$. Defined literals that have no defining core and therefore no defining literal are literals that are trivially false or true. Consider, for example, $g(f(a)) \approx g(f(a))$. This literal is trivially true in Γ. Thus an empty subset of Γ is sufficient to show that $g(f(a)) \approx g(f(a))$ is defined in Γ.

Definition 6 *(Literal Level)* Let Γ be a trail. A ground literal $L \in \Gamma$ is of *level* i if L is annotated with i in Γ. A defined ground literal $L \notin \Gamma$ is of level i if the defining literal of L is of level i. If L has no defining literal, then L is of level 0. A ground clause D is of level i if i is the maximum level of a literal in D.

The restriction to minimal subsequences for the defining literal and definition of a level eventually guarantee that learned clauses are smaller in the trail ordering. This enables completeness in combination with learning non-redundant clauses as shown later.

Lemma 1 *Let Γ_1 be a trail and K a defined literal that is of level i in Γ_1. Then K is of level i in a trail $\Gamma{:=}\Gamma_1, \Gamma_2$.*

Definition 7 Let Γ be a trail and $L \in \Gamma$ a literal. L is called a *decision literal* if $\Gamma = \Gamma_0, K^{i:C\cdot\tau}, L^{i+1:C'\cdot\tau'}, \Gamma_1$. Otherwise L is called a *propagated literal*.

In other words: L is a decision literal if the level of L is one greater than the level of the preceeding literal K. In our above example $g(f(a)) \approx g(f(c))$ is of level 3 since the defining literal $b \approx c$ is annotated with 3. $a \not\approx b$ on the other hand is of level 2.

We define a well-founded total strict ordering which is induced by the trail and with which non-redundancy is proven in Sect. 4. Unlike SCL [15, 21] we use this ordering for the inference rules as well. In previous SCL calculi, conflict resolution automatically chooses the greatest literal and resolves with this literal. In SCL(EQ) this is generalized. Coming back to our running example above, suppose we have a conflict clause $f(b) \not\approx f(c) \vee b \not\approx c$. The defining literal for both inequations is $b \approx c$. So we could do paramodulation inferences with both literals. The following ordering makes this non-deterministic choice deterministic.

Definition 8 *(Trail Induced Ordering)* Let $\Gamma{:=}[L_1^{i_1:C_1\cdot\sigma_1}, ..., L_n^{i_n:C_n\cdot\sigma_n}]$ be a trail, β a ground term such that $\{L_1, ..., L_n\} \prec_T \beta$ and $M_{i,j}$ all β-defined ground literals not contained in $\Gamma \cup comp(\Gamma)$: for a defining literal $max_\Gamma(M_{i,j}) = L_i$ and for two literals $M_{i,j}$, $M_{i,k}$ we have $j < k$ if $M_{i,j} \prec_T M_{i,k}$.

The trail induces a total well-founded strict order $\prec_{\Gamma^*}$ on *β-defined* ground literals $M_{k,l}$, $M_{m,n}$, L_i, L_j of level greater than zero, where

1. $M_{i,j} \prec_{\Gamma^*} M_{k,l}$ if $i < k$ or ($i = k$ and $j < l$);
2. $L_i \prec_{\Gamma^*} L_j$ if $L_i \prec_\Gamma L_j$;
3. $comp(L_i) \prec_{\Gamma^*} L_j$ if $L_i \prec_\Gamma L_j$;
4. $L_i \prec_{\Gamma^*} comp(L_j)$ if $L_i \prec_\Gamma L_j$ or $i = j$;
5. $comp(L_i) \prec_{\Gamma^*} comp(L_j)$ if $L_i \prec_\Gamma L_j$;
6. $L_i \prec_{\Gamma^*} M_{k,l}$, $\text{comp}(L_i) \prec_{\Gamma^*} M_{k,l}$ if $i \leq k$;

7. $M_{k,l} \prec_{\Gamma^*} L_i$, $M_{k,l} \prec_{\Gamma^*} \text{comp}(L_i)$ if $k < i$;

and for all β-*defined* literals L of level zero:

8. $\prec_{\Gamma^*} := \prec_T$;
9. $L \prec_{\Gamma^*} K$ if K is of level greater than zero and K is β-*defined*;

and can eventually be extended to β-*undefined* ground literals K, H by

10. $K \prec_{\Gamma^*} H$ if $K \prec_T H$;
11. $L \prec_{\Gamma^*} H$ if L is β-*defined*.

The literal ordering $\prec_{\Gamma^*}$ is extended to ground clauses by multiset extension and identified with $\prec_{\Gamma^*}$ as well.

Note, that in the above definition for a given i we can always put the $M_{i,j}$ in an order corresponding to the integers. This is due to the fact, that the total number of ground literals is countable and since there are only finitely many β-defined literals.

Lemma 2 *(Properties of $\prec_{\Gamma^*}$)*

1. $\prec_{\Gamma^*}$ *is well-defined.*
2. $\prec_{\Gamma^*}$ *is a total strict order, i.e.* $\prec_{\Gamma^*}$ *is irreflexive, transitive and total.*
3. $\prec_{\Gamma^*}$ *is a well-founded ordering.*

Example 1 Assume a trail $\Gamma := [a \approx b^{1:C_0 \cdot \sigma_0}, c \approx d^{1:C_1 \cdot \sigma_1}, f(a') \not\approx f(b')^{1:C_2 \cdot \sigma_2}]$, select KBO as the term ordering $\prec_T$ where all symbols have weight one and $a \prec a' \prec b \prec b' \prec c \prec d \prec f$ and a ground term $\beta := f(f(a))$. According to the trail induced ordering we have that $a \approx b \prec_{\Gamma^*} c \approx d \prec_{\Gamma^*} f(a') \not\approx f(b')$ by 8.2. Furthermore we have that

$$a \approx b \prec_{\Gamma^*} a \not\approx b \prec_{\Gamma^*} c \approx d \prec_{\Gamma^*} c \not\approx d \prec_{\Gamma^*} f(a') \not\approx f(b') \prec_{\Gamma^*} f(a') \approx f(b'),$$

by 8.3 and 8.4. Now for any literal L that is β-*defined* in Γ and the defining literal is $a \approx b$ it holds that $a \not\approx b \prec_{\Gamma^*} L \prec_{\Gamma^*} c \approx d$ by 8.6 and 8.7. This holds analogously for all literals that are β-*defined* in Γ and the defining literal is $c \approx d$ or $f(a') \not\approx f(b')$. Thus we get:

$$\begin{aligned} L_1 \prec_{\Gamma^*} \ldots \prec_{\Gamma^*} a \approx b \prec_{\Gamma^*} a \not\approx b \prec_{\Gamma^*} f(a) \approx f(b) \prec_{\Gamma^*} f(a) \not\approx f(b) \prec_{\Gamma^*} \\ c \approx d \prec_{\Gamma^*} c \not\approx d \prec_{\Gamma^*} f(c) \approx f(d) \prec_{\Gamma^*} f(c) \not\approx f(d) \prec_{\Gamma^*} \\ f(a') \not\approx f(b') \prec_{\Gamma^*} f(a') \approx f(b') \prec_{\Gamma^*} a' \approx b' \prec_{\Gamma^*} a' \not\approx b' \prec_{\Gamma^*} K_1 \prec_{\Gamma^*} \ldots \end{aligned}$$

where K_i are the β-*undefined* literals and L_j are the trivially defined literals.

Table 1 summarizes the various orders presented so far.

Definition 9 *(Rewrite Step)* A *rewrite step* is a five-tuple $(s\#t \cdot \sigma, s\#t \vee C \cdot \sigma, R, S, p)$ and inductively defined as follows. The tuple $(s\#t \cdot \sigma, s\#t \vee C \cdot \sigma, \epsilon, \epsilon, \epsilon)$ is a rewrite step. Given rewrite steps R, S and a position p then $(s\#t \cdot \sigma, s\#t \vee C \cdot \sigma, R, S, p)$ is a *rewrite step*. The literal $s\#t$ is called the *rewrite literal*. In case R, S are not ϵ, the rewrite literal of R is an equation.

Note that R and S in the above definition describe the "history" of a rewrite step, i.e. they contain all preceeding rewrite steps.

Rewriting is one of the core features of our calculus. The following definition describes a rewrite inference between two clauses. Note that unlike the superposition calculus we allow rewriting below variable level.

Table 1 Summary of the orderings presented so far

Order		Description
Term order	$\prec_T$	Well-founded rewrite ordering on terms
		Total on ground terms
		For all ground terms t there exist only finitely many ground terms $s \prec_T t$
Trail order	$\prec_\Gamma$	Sequence ordering given by the trail Γ
Trail induced order	$\prec_{\Gamma*}$	Extends the sequence ordering to all literals implicitly defined due to literals of level >0
		Uses $\prec_T$ for literals of level 0 and undefined literals
		Literals of level 0 are smaller and undefined literals are greater than literals of level >0

Definition 10 *(Rewrite Inference)* Let $I_1{:=}(l_1 \approx r_1 \cdot \sigma_1, l_1 \approx r_1 \vee C_1 \cdot \sigma_1, R_1, L_1, p_1)$ and $I_2{:=}(l_2\#r_2 \cdot \sigma_2, l_2\#r_2 \vee C_2 \cdot \sigma_2, R_2, L_2, p_2)$ be two variable disjoint rewrite steps where $r_1\sigma_1 \prec_T l_1\sigma_1$, $(l_2\#r_2)\sigma_2|_p = l_1\sigma_1$ for some position p. We distinguish two cases:

1. if $p \in \text{pos}(l_2\#r_2)$ and $\mu{:=}\,\text{mgu}((l_2\#r_2)|_p, l_1)$ then $(((l_2\#r_2)[r_1]_p)\mu \cdot \sigma_1\sigma_2, ((l_2\#r_2)[r_1]_p)\mu \vee C_1\mu \vee C_2\mu \cdot \sigma_1\sigma_2, I_1, I_2, p)$ is the result of a rewrite inference.
2. if $p \notin \text{pos}(l_2\#r_2)$ then let $(l_2\#r_2)\delta$ be the most general instance of $l_2\#r_2$ such that $p \in \text{pos}((l_2\#r_2)\delta)$, δ introduces only fresh variables and $(l_2\#r_2)\delta\sigma_2\rho = (l_2\#r_2)\sigma_2$ for some minimal ρ. Let $\mu{:=}\,\text{mgu}((l_2\#r_2)\delta|_p, l_1)$. Then $((l_2\#r_2)\delta[r_1]_p\mu \cdot \sigma_1\sigma_2\rho, (l_2\#r_2)\delta[r_1]_p\mu \vee C_1\mu \vee C_2\delta\mu \cdot \sigma_1\sigma_2\rho, I_1, I_2, p)$ is the result of a rewrite inference.

Note that case 1 describes rewriting above or at a variable and case 2 describes rewriting inside a variable.

Lemma 3 *Let* $I_1{:=}(l_1 \approx r_1 \cdot \sigma_1, l_1 \approx r_1 \vee C_1 \cdot \sigma_1, R_1, L_1, p_1)$ *and* $I_2{:=}(l_2\#r_2 \cdot \sigma_2, l_2\#r_2 \vee C_2 \cdot \sigma_2, R_2, L_2, p_2)$ *be two variable disjoint rewrite steps where* $r_1\sigma_1 \prec_T l_1\sigma_1$, $(l_2\#r_2)\sigma_2|_p = l_1\sigma_1$ *for some position* p. *Let* $I_3{:=}(l_3\#r_3 \cdot \sigma_3, l_3\#r_3 \vee C_3 \cdot \sigma_3, I_1, I_2, p)$ *be the result of a rewrite inference. Then:*

1. $C_3\sigma_3 = (C_1 \vee C_2)\sigma_1\sigma_2$ *and* $l_3\#r_3\sigma_3 = (l_2\#r_2)\sigma_2[r_1\sigma_1]_p$.
2. $(l_3\#r_3)\sigma_3 \prec_T (l_2\#r_2)\sigma_2$
3. *If* $N \models (l_1 \approx r_1 \vee C_1) \wedge (l_2\#r_2 \vee C_2)$ *for some set of clauses* N, *then* $N \models l_3\#r_3 \vee C_3$

Now that we have defined rewrite inferences we can use them to define a *reduction chain application* and a *refutation*, which are sequences of rewrite steps. Intuitively speaking, a *reduction chain application* reduces a literal in a clause with literals in $conv(\Gamma)$ until it is irreducible. A *refutation* for a literal L that is β*-false* in Γ for a given β, is a sequence of rewrite steps with literals in Γ, L such that $\bot$ is inferred. Refutations for the literals of the conflict clause will be examined during conflict resolution by the rule Explore-Refutation.

Definition 11 *(Reduction Chain)* Let Γ be a trail. A *reduction chain* $\mathcal{P}$ from Γ is a sequence of rewrite steps $[I_1, ..., I_m]$ such that for each $I_i = (s_i\#t_i \cdot \sigma_i, s_i\#t_i \vee C_i \cdot \sigma_i, I_j, I_k, p_i)$ either

1. $s_i\#t_i^{n_i : s_i\#t_i \vee C_i \cdot \sigma}$ is contained in Γ and $I_j = I_k = p_i = \epsilon$ or
2. I_i is the result of a rewriting inference from rewrite steps I_j, I_k out of $[I_1, ..., I_m]$ where $j, k < i$.

Let $(l \# r)\delta^{o:l\#r\vee C\cdot\delta}$ be an annotated ground literal. A *reduction chain application* from Γ to $l \# r$ is a reduction chain $[I_1, ..., I_m]$ from Γ, $(l \# r)\delta^{o:l\#r\vee C\cdot\delta}$ such that $l\delta\downarrow_{\text{conv}(\Gamma)} = s_m\sigma_m$ and $r\delta\downarrow_{\text{conv}(\Gamma)} = t_m\sigma_m$. We assume reduction chain applications to be minimal, i.e., if any rewrite step is removed from the sequence it is no longer a reduction chain application.

Definition 12 *(Refutation)* Let Γ be a trail and $(l \# r)\delta^{o:l\#r\vee C\cdot\delta}$ an annotated ground literal that is *β-false* in Γ for a given β. A *refutation* $\mathcal{P}$ from Γ and $l \# r$ is a reduction chain $[I_1, ..., I_m]$ from Γ, $(l \# r)\delta^{o:l\#r\vee C\cdot\delta}$ such that $(s_m\#t_m)\sigma_m = s \not\approx s$ for some s. We assume refutations to be minimal, i.e., if any rewrite step $I_k, k < m$ is removed from the refutation, it is no longer a refutation.

3.1 The SCL(EQ) Inference Rules

We can now define the rules of our calculus based on the previous definitions. A *state* is a six-tuple $(\Gamma; N; U; \beta; k; D)$ similar to the SCL calculus, where Γ a sequence of annotated ground literals, N and U the sets of initial and learned clauses, β is a ground term such that for all $L \in \Gamma$ it holds $L \prec_T \beta$, k is the decision level, and D a status that is $\top$, $\bot$ or a closure $C \cdot \sigma$. Before we propagate or decide any literal, we make sure that it is irreducible in the current trail. Together with the design of $\prec_{\Gamma^*}$ this eventually enables rewriting as a simplification rule.

Propagate

$$(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL(EQ)}} (\Gamma, s_m\#t_m\sigma_m^{k:(s_m\#t_m\vee C_m)\cdot\sigma_m}; N; U; \beta; k; \top)$$

provided there is a $C \in (N \cup U)$, σ grounding for C, $C = C_0 \vee C_1 \vee L$, $\Gamma \models \neg C_0\sigma$, $C_1\sigma = L\sigma \vee ... \vee L\sigma, C_1 = L_1 \vee ... \vee L_n, \mu = mgu(L_1, ..., L_n, L)$ $L\sigma$ is *β-undefined* in Γ, $(C_0 \vee L)\mu\sigma \prec_T \beta, \sigma$ is irreducible by $conv(\Gamma)$, $[I_1, \ldots, I_m]$ is a reduction chain application from Γ to $L\sigma^{k:(L\vee C_0)\mu\cdot\sigma}$ where $I_m = (s_m\#t_m \cdot \sigma_m, s_m\#t_m \vee C_m \cdot \sigma_m, I_j, I_k, p_m)$.

The rule Propagate finds a ground instance of a clause which is propagable, i.e. it contains (possibly multiple occurences of) a literal that is *undefined* and all other literals are *false* in the trail. The multiple occurences of the undefined literal are factored. Then it adds the normal form of this literal to the trail. The propagating clause is reduced by the corresponding paramodulation steps. Note that the definition of Propagate also includes the case where $L\sigma$ is irreducible by Γ. In this case $L = s_m\#t_m$ and $m = 1$. The rule Decide below, is similar to Propagate, except for the subclause C_0 which must be *β-undefined* or *β-true* in Γ, i.e., Propagate cannot be applied and the decision literal is annotated by a tautology.

Decide

$$(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL(EQ)}} (\Gamma, s_m\#t_m\sigma_m^{k+1:(s_m\#t_m\vee comp(s_m\#t_m))\cdot\sigma_m}; N; U; \beta; k + 1; \top)$$

provided there is a $C \in (N \cup U)$, σ grounding for C, $C = C_0 \vee L$, $C_0\sigma$ is *β-undefined* or *β-true* in Γ, $L\sigma$ is *β-undefined* in Γ, $(C_0 \vee L)\sigma \prec_T \beta$, σ is irreducible by $conv(\Gamma)$, $[I_1, \ldots, I_m]$ is a reduction chain application from Γ to $L\sigma^{k+1:L\vee C_0\cdot\sigma}$ where $I_m = (s_m\#t_m \cdot \sigma_m, s_m\#t_m \vee C_m \cdot \sigma_m, I_j, I_k, p_m)$.

Conflict $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL(EQ)}} (\Gamma; N; U; \beta; k; D)$

provided there is a $D' \in (N \cup U)$, σ grounding for D', $D'\sigma$ is *β-false* in Γ, σ is irreducible by $conv(\Gamma)$, $D = \bot$ if $D'\sigma$ is of level 0 and $D = D' \cdot \sigma$ otherwise.

For the non-equational case, when a conflict clause is found by an SCL calculus [15, 21], the complements of its first-order ground literals are contained in the trail. For equational literals this is not the case, in general. The proof showing D to be *β-false* with respect to Γ is a rewrite proof with respect to conv(Γ). This proof needs to be analyzed to eventually

perform paramodulation steps on D or to replace D by a $\prec_{\Gamma^*}$ smaller β-*false* clause showing up in the proof.

Skip $(\Gamma, K^{l:C\cdot\tau}, L^{k:C'\cdot\tau'}; N; U; \beta; k; D\cdot\sigma) \Rightarrow_{\text{SCL(EQ)}} (\Gamma, K^{l:C\cdot\tau}; N; U; \beta; l; D\cdot\sigma)$ if $D\sigma$ is β-*false* in $\Gamma, K^{l:C\cdot\tau}$.

The Explore-Refutation rule is the FOL with Equality counterpart to the resolve rule in CDCL or SCL. While in CDCL or SCL complementary literals of the conflict clause are present on the trail and can directly be used for resolution steps, this needs a generalization for FOL with Equality. Here, in general, we need to look at (rewriting) refutations of the conflict clause and pick an appropriate clause from the refutation as the next conflict clause.

Explore-Refutation

$(\Gamma, L; N; U; \beta; k; (D \vee s \# t)\cdot\sigma)) \Rightarrow_{\text{SCL(EQ)}} (\Gamma, L; N; U; \beta; k; (s_j \# t_j \vee C_j)\cdot\sigma_j)$ if $(s \# t)\sigma$ is strictly $\prec_{\Gamma^*}$ maximal in $(D \vee s \# t)\sigma$, L is the defining literal of $(s \# t)\sigma$, $[I_1, ..., I_m]$ is a refutation from Γ and $(s \# t)\sigma$, $I_j = (s_j \# t_j \cdot \sigma_j, (s_j \# t_j \vee C_j)\cdot\sigma_j, I_l, I_k, p_j)$, $1 \leq j \leq m$, $(s_j \# t_j \vee C_j)\sigma_j \prec_{\Gamma^*} (D \vee s \# t)\sigma$, $(s_j \# t_j \vee C_j)\sigma_j$ is β-*false* in Γ.

Factorize

$(\Gamma; N; U; \beta; k; (D \vee L \vee L')\cdot\sigma) \Rightarrow_{\text{SCL(EQ)}} (\Gamma; N; U; \beta; k; (D \vee L)\mu\cdot\sigma)$ provided $L\sigma = L'\sigma$, and $\mu = \text{mgu}(L, L')$.

Equality-Resolution

$(\Gamma; N; U; \beta; k; (D \vee s \not\approx s')\cdot\sigma) \Rightarrow_{\text{SCL(EQ)}} (\Gamma; N; U; \beta; k; D\mu\cdot\sigma)$ provided $s\sigma = s'\sigma$, $\mu = mgu(s, s')$.

For backtracking we have to make sure, that the learned clause is not *false* in the resulting trail. It is not sufficient to backtrack to the point where the clause with the current substitution is no longer false, but where it is no longer false with all possible substitutions.

Backtrack $(\Gamma, K, \Gamma'; N; U; \beta; k; (D \vee L)\cdot\sigma) \Rightarrow_{\text{SCL(EQ)}} (\Gamma; N; U \cup \{D \vee L\}; \beta; j - i; \top)$ provided $D\sigma$ is of level i' where $i' < k$, K is of level j and Γ, K the minimal trail subsequence such that there is a grounding substitution τ with $(D \vee L)\tau$ β-false in Γ, K but not in Γ; $i = 1$ if K is a decision literal and $i = 0$ otherwise.

Grow $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL(EQ)}} (\epsilon; N; U; \beta'; 0; \top)$ provided $\beta \prec_T \beta'$.

In addition to soundness and completeness of the SCL(EQ) rules their tractability in practice is an important property for a successful implementation. In particular, finding propagating literals or detecting a false clause under some grounding. It turns out that these operations are NP-complete, similar to first-order subsumption which has been shown to be tractable in practice.

Lemma 4 *Assume that all ground terms t with $t \prec_T \beta$ for any β are polynomial in the size of β. Then testing Propagate (Conflict) is NP-Complete, i.e., the problem of checking for a given clause C whether there exists a grounding substitution σ such that $C\sigma$ propagates (is false) is NP-Complete.*

Example 2 (SCL(EQ) vs. Superposition: Saturation) Consider the following clauses:

$$N{:=}\{C_1{:=}c \approx d \vee D,\ C_2{:=}a \approx b \vee c \not\approx d,\ C_3{:=}f(a) \not\approx f(b) \vee g(c) \not\approx g(d)\},$$

where again we assume a KBO with all symbols having weight one, precedence $d \prec c \prec b \prec a \prec g \prec f$ and $\beta{:=}f(f(g(a)))$. Suppose that we first decide $c \approx d$ and then propagate $a \approx b$: $\Gamma = [c \approx d^{1:c\approx d \vee c\not\approx d}, a \approx b^{1:C_2}]$. Now we have a conflict with C_3. Explore-Refutation applied to the conflict clause C_3 results in a paramodulation inference between

C_3 and C_2. Another application of Equality-Resolution gives us the new conflict clause $C_4{:=}c \not\approx d \vee g(c) \not\approx g(d)$. Now we can Skip the last literal on the trail, which gives us $\Gamma = [c \approx d^{1:c\approx d \vee c \not\approx d}]$. Another application of the Explore-Refutation rule to C_4 using the decision justification clause followed by Equality-Resolution and Factorize gives us $C_5{:=}c \not\approx d$. Thus with SCL(EQ) the following clauses remain:

$$\begin{array}{c} C_1' = D \qquad C_5 = c \not\approx d \\ C_3 = f(a) \not\approx f(b) \vee g(c) \not\approx g(d) \end{array}$$

where we derived C_1' out of C_1 by subsumption resolution [37] using C_5. Actually, subsumption resolution is compatible with the general redundancy notion of SCL(EQ), see Lemma 6. Now we consider the same example with superposition and the very same ordering (N_i is the clause set of the previous step and N_0 the initial clause set N).

$$\begin{array}{l} N_0 \Rightarrow_{Sup(C_2,C_3)} N_1 \cup \{C_4{:=}c \not\approx d \vee g(c) \not\approx g(d)\} \\ \quad \Rightarrow_{Sup(C_1,C_4)} N_2 \cup \{C_5{:=}c \not\approx d \vee D\} \Rightarrow_{Sup(C_1,C_5)} N_3 \cup \{C_6{:=}D\} \end{array}$$

Thus superposition ends up with the following clauses:

$$\begin{array}{ll} C_2 = a \approx b \vee c \not\approx d & C_3 = f(a) \not\approx f(b) \vee g(c) \not\approx g(d) \\ C_4 = c \not\approx d \vee g(c) \not\approx g(d) & C_6 = D \end{array}$$

The superposition calculus generates more and larger clauses.

Example 3 (SCL(EQ) vs. Superposition: Refutation) Suppose the following set of clauses: $N{:=}\{C_1{:=}f(x) \not\approx a \vee f(x) \approx b, C_2{:=}f(f(y)) \approx y, C_3{:=}a \not\approx b\}$ where again we assume a KBO with all symbols having weight one, precedence $b \prec a \prec f$ and $\beta{:=}f(f(f(a)))$. A long refutation by the superposition calculus results in the following (N_i is the clause set of the previous step and N_0 the initial clause set N):

$$\begin{array}{l} N_0 \Rightarrow_{Sup(C_1,C_2)} N_1 \cup \{C_4{:=}y \not\approx a \vee f(f(y)) \approx b\} \\ \quad \Rightarrow_{Sup(C_1,C_4)} N_2 \cup \{C_5{:=}a \not\approx b \vee f(f(y)) \approx b \vee y \not\approx a\} \\ \quad \Rightarrow_{Sup(C_2,C_5)} N_3 \cup \{C_6{:=}a \not\approx b \vee b \approx y \vee y \not\approx a\} \\ \quad \Rightarrow_{Sup(C_2,C_4)} N_4 \cup \{C_7{:=}y \approx b \vee y \not\approx a\} \\ \quad \Rightarrow_{EqRes(C_7)} N_5 \cup \{C_8{:=}a \approx b\} \Rightarrow_{Sup(C_3,C_8)} N_6 \cup \{\bot\} \end{array}$$

The shortest refutation by the superposition calculus is as follows:

$$\begin{array}{l} N_0 \Rightarrow_{Sup(C_1,C_2)} N_1 \cup \{C_4{:=}y \not\approx a \vee f(f(y)) \approx b\} \\ \quad \Rightarrow_{Sup(C_2,C_4)} N_2 \cup \{C_5{:=}y \approx b \vee y \not\approx a\} \\ \quad \Rightarrow_{EqRes(C_5)} N_3 \cup \{C_6{:=}a \approx b\} \Rightarrow_{Sup(C_3,C_6)} N_4 \cup \{\bot\} \end{array}$$

In SCL(EQ) on the other hand we would always first propagate $a \not\approx b$, $f(f(a)) \approx a$ and $f(f(b)) \approx b$. As soon as $a \not\approx b$ and $f(f(a)) \approx a$ are propagated we have a conflict with $C_1\{x \to f(a)\}$. So suppose in the worst case we propagate:

$$\Gamma{:=}[a \not\approx b^{0:a\not\approx b} f(f(b)) \approx b^{0:(f(f(y))\approx y)\{y\to b\}} f(f(a)) \approx a^{0:(f(f(y))\approx y)\{y\to a\}}]$$

Now we have a conflict with $C_1\{x \to f(a)\}$. Since there is no decision literal on the trail, *Conflict* rule immediately returns $\bot$ and we are done.

4 Soundness and Completeness

In this section we show soundness and refutational completeness of SCL(EQ) under the assumption of a regular run. We provide the definition of a regular run and show that for a regular run all learned clauses are non-redundant according to our trail induced ordering. We start with the definition of a sound state.

Definition 13 A state $(\Gamma; N; U; \beta; k; D)$ is sound if the following conditions hold:

1. Γ is a consistent sequence of annotated literals,
2. for each decomposition $\Gamma = \Gamma_1, L\sigma^{i:(C \vee L)\cdot\sigma}, \Gamma_2$ where $L\sigma$ is a propagated literal, we have that $C\sigma$ is β-*false* in Γ_1, $L\sigma$ is β-*undefined* in Γ_1 and irreducible by $conv(\Gamma_1)$, $N \cup U \models (C \vee L)$ and $(C \vee L)\sigma \prec_T \beta$,
3. for each decomposition $\Gamma = \Gamma_1, L\sigma^{i:(L \vee comp(L))\cdot\sigma}, \Gamma_2$ where $L\sigma$ is a decision literal, we have that $L\sigma$ is β-*undefined* in Γ_1 and irreducible by $conv(\Gamma_1)$, $N \cup U \models (L \vee comp(L))$ and $(L \vee comp(L))\sigma \prec_T \beta$,
4. $N \models U$,
5. if $D = C \cdot \sigma$, then $C\sigma$ is β-*false* in Γ, $N \cup U \models C$

Lemma 5 *The initial state* $(\epsilon; N; \emptyset; \beta; 0; \top)$ *is sound.*

Definition 14 A run is a sequence of applications of SCL(EQ) rules starting from the initial state.

Theorem 1 *Assume a state* $(\Gamma; N; U; \beta; k; D)$ *resulting from a run. Then* $(\Gamma; N; U; \beta; k; D)$ *is sound.*

Next, we give the definition of a regular run. Intuitively speaking, in a regular run we are always allowed to do decisions except if

1. a literal can be propagated before the first decision and
2. the negation of a literal can be propagated.

To ensure non-redundant learning we enforce at least one application of Skip during conflict resolution except for the special case of a conflict after a decision.

Definition 15 *(Regular Run)* A run is called *regular* if

1. the rules *Conflict* and *Factorize* have precedence over all other rules,
2. If $k = 0$ in a state $(\Gamma; N; U; \beta; k; D)$, then *Propagate* has precedence over *Decide*,
3. If an annotated literal $L^{k:C\cdot\sigma}$ could be added by an application of Propagate on Γ in a state $(\Gamma; N; U; \beta; k; D)$ and $C \in N \cup U$, then the annotated literal $comp(L)^{k+1:C'\cdot\sigma'}$ is not added by Decide on Γ,
4. during conflict resolution *Skip* is applied at least once, except if *Conflict* is applied immediately after an application of *Decide*.
5. if *Conflict* is applied immediately after an application of *Decide*, then Backtrack is only applied in a state $(\Gamma, L'; N; U; \beta; k; D \cdot \sigma)$ if $L\sigma = comp(L')$ for some $L \in D$.

Now we show that any learned clause in a regular run is non-redundant according to our trail induced ordering.

Lemma 6 (Non-Redundant Clause Learning) *Let N be a clause set. The clauses learned during a regular run in SCL(EQ) are not redundant with respect to $\prec_{\Gamma^*}$ and $N \cup U$. For the trail only non-redundant clauses need to be considered.*

The proof of Lemma 6 is based on the fact that conflict resolution eventually produces a clause smaller then the original conflict clause with respect to $\prec_{\Gamma^*}$. All simplifications, e.g., contextual rewriting, as defined in [2, 24, 37, 38, 40, 41], are therefore compatible with Lemma 6 and may be applied to the newly learned clause as long as they respect the induced trail ordering. In detail, let Γ be the trail before the application of rule Backtrack. The newly learned clause can be simplified according to the induced trail ordering $\prec_{\Gamma^*}$ as long as the simplified clause is smaller with respect to $\prec_{\Gamma^*}$.

Another important consequence of Lemma 6 is that newly learned clauses need not to be considered for redundancy. Furthermore, the SCL(EQ) calculus always terminates, Lemma 11, because there only finitely many non-redundant clauses with respect to a fixed β.

For dynamic redundancy, we have to consider the fact that the induced trail ordering changes. At this level, only redundancy criteria and simplifications that are compatible with *all* induced trail orderings may be applied. Due to the construction of the induced trail ordering, it is compatible with $\prec_T$ for unit clauses.

Lemma 7 (Unit Rewriting) *Assume a state* $(\Gamma; N; U; \beta; k; D)$ *resulting from a regular run where the current level* $k > 0$ *and a unit clause* $l \approx r \in N$. *Now assume a clause* $C \vee L[l']_p \in N$ *such that* $l' = l\mu$ *for some matcher* μ. *Now assume some arbitrary grounding substitutions* σ' *for* $C \vee L[l']_p$, σ *for* $l \approx r$ *such that* $l\sigma = l'\sigma'$ *and* $r\sigma \prec_T l\sigma$. *Then* $(C \vee L[r\mu\sigma\sigma']_p)\sigma' \prec_{\Gamma^*} (C \vee L[l']_p)\sigma'$.

In addition, any notion that is based on a literal subset relationship is also compatible with ordering changes. The standard example is subsumption.

Lemma 8 *Let* C, D *be two clauses. If there exists a substitution* σ *such that* $C\sigma \subset D$, *then* D *is redundant with respect to* C *and any* $\prec_{\Gamma^*}$.

The notion of redundancy, Definition 1, only supports a strict subset relation for Lemma 8, similar to the superposition calculus. However, the newly generated clauses of SCL(EQ) are the result of paramodulation inferences [32]. In a recent contribution to dynamic, abstract redundancy [36] it is shown that also the non-strict subset relation in Lemma 8, i.e., $C\sigma \subseteq D$, preserves completeness.

If all stuck states, see below Definition 16, with respect to a fixed β are visited before increasing β then this provides a simple dynamic fairness strategy.

When unit reduction or any other form of supported rewriting is applied to clauses smaller than the current β, it can be applied independently from the current trail. If, however, unit reduction is applied to clauses larger than the current β then the calculus must do a restart to its initial state, in particular the trail must be emptied, as for otherwise rewriting may result generating a conflict that did not exist with respect to the current trail before the rewriting. This is analogous to a restart in CDCL once a propositional unit clause is derived and used for simplification. More formally, we add the following new Restart rule to the calculus to reset the trail to its initial state after a unit reduction.

Restart $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL(EQ)}} (\epsilon; N; U; \beta; 0; \top)$

Next we show refutation completeness of SCL(EQ). To achieve this we first give a definition of a stuck state. Then we show that stuck states only occur if all ground literals $L \prec_T \beta$ are β-*defined* in Γ and not during conflict resolution. Finally we show that conflict resolution will always result in an application of Backtrack. This allows us to show termination (without application of Grow) and refutational completeness.

Definition 16 *(Stuck State)* A state $(\Gamma; N; U; \beta; k; D)$ is called *stuck* if $D \neq \bot$ and none of the rules of the calculus, except for Grow, is applicable.

Lemma 9 (Form of Stuck States) *If a regular run (without rule Grow) ends in a stuck state* $(\Gamma; N; U; \beta; k; D)$*, then* $D = \top$ *and all ground literals* $L\sigma \prec_T \beta$*, where* $L \vee C \in (N \cup U)$ *are* β*-defined in* Γ.

Lemma 10 *Suppose there is a sound state* $(\Gamma; N; U; \beta; k; D)$ *resulting from a regular run where* $D \notin \{\top, \bot\}$*. If* `Backtrack` *is not applicable then any set of applications of Explore-Refutation,* `Skip`*, Factorize, Equality-Resolution will finally result in a sound state* $(\Gamma'; N; U; \beta; k; D')$*, where* $D' \prec_{\Gamma^*} D$*. Then Backtrack will be finally applicable.*

Corollary 1 (Satisfiable Clause Sets) *Let N be a satisfiable clause set. Then any regular run without rule Grow will end in a stuck state, for any* β.

Thus a stuck state can be seen as an indication for a satisfiable clause set. Of course, it remains to be investigated whether the clause set is actually satisfiable. Superposition is one of the strongest approaches to detect satisfiability and constitutes a decision procedure for many decidable first-order fragments [3, 23]. Now given a stuck state and some specific ordering such as KBO, LPO, or some polynomial ordering [20], it is decidable whether the ordering can be instantiated from a stuck state such that Γ coincides with the superposition model operator on the ground terms smaller than β. In this case it can be effectively checked whether the clauses derived so far are actually saturated by the superposition calculus with respect to this specific ordering. In this sense, SCL(EQ) has the same power to decide satisfiability of first-order clause sets as superposition.

Definition 17 A regular run terminates in a state $(\Gamma; N; U; \beta; k; D)$ if $D = \top$ and no rule is applicable, or $D = \bot$.

Lemma 11 *Let N be a set of clauses and* β *be a ground term. Then any regular run that never uses Grow terminates.*

Lemma 12 *If a regular run reaches the state* $(\Gamma; N; U; \beta; k; \bot)$ *then N is unsatisfiable.*

Theorem 2 (Refutational Completeness) *Let N be an unsatisfiable clause set, and* $\prec_T$ *a desired term ordering. For any ground term* β *where* $gnd_{\prec_T \beta}(N)$ *is unsatisfiable, any regular SCL(EQ) run without rule Grow will terminate by deriving* $\bot$.

5 Discussion

We presented SCL(EQ), a new sound and complete calculus for reasoning in first-order logic with equality. We will now discuss some of its aspects and present ideas for future work beyond the scope of this paper.

The SCL(EQ) calculus can be viewed as a generalization of the first-order without equality SCL calculus [17], where syntactic equality with respect to trail literals is replaced with equality modulo the presented equational theory. If standard first-order literals like $R(x, y)$ are represented by equations like $f_R(x, y) \approx \texttt{true}$ then performing SCL(EQ) on the latter simplifies to classical SCL reasoning on the first-order literals with a slightly different strategy.

The trail induced ordering, Definition 8, is the result of letting the calculus follow the logical structure of the clause set on the literal level and at the same time supporting rewriting

at the term level. It can already be seen by examples on ground clauses over (in)equations over constants that this combination requires a layered approach as suggested by Definition 8, see Example 4.

Example 4 (Propagate Smaller Equation) Assume a term ordering $\prec_{kbo}$, unique weight 1 and with precedence $d \prec c \prec b \prec a$. Further assume β to be large enough. Assume the ground clause set N solely built out of constants

$$C_1{:=}c \approx d \qquad C_2{:=}c \not\approx d \vee a \approx b$$
$$C_3{:=}a \not\approx b \vee a \approx c$$

and the trail $\Gamma{:=}[c \approx d^{0:C_1}, a \approx b^{0:C_2}, b \approx d^{0:C_3}]$. Now, although the first two steps propagated equations that are strictly maximal in the ordering in their respective clauses, the finally propagated equation $b \approx d$ is smaller in the term ordering $\prec_{kbo}$ than $a \approx b$. Thus the structure of the clause set forces propagation of a smaller equation in the term ordering. So the more complicated trail ordering is a result of following the structure of the clause set rather than employing an a priori fixed ordering.

In case the calculus runs into a stuck state, i.e., the current trail is a model for the set of considered ground instances, then the trail information can be effectively used for a guided continuation. For example, in order to use the trail to certify a model, the trail literals can be used to guide the design of a lifted ordering for the clauses with variables such that propagated trail literals are maximal in respective clauses. Then it could be checked by superposition, if the current clause is saturated by such an ordering. If this is not the case, then there must be a superposition inference larger than the current β, thus giving a hint on how to extend β. Another possibility is to try to extend the finite set of ground terms considered in a stuck state to the infinite set of all ground terms by building extended equivalence classes following patterns that ensure decidability of clause testing, similar to the ideas in [15]. If this fails, then again this information can be used to find an appropriate extension term β for rule Grow.

In contrast to superposition, SCL(EQ) does also inferences below variable level. In general, single superposition inferences below variables are redundant [2]. Inferences in SCL(EQ) are guided by a false clause with respect to a partial model assumption represented by the trail. They are typically not single superposition steps, but a sequence of superposition inferences eventually resulting in a non-redundant clause changing the partial model assumption. Therefore, compared to the syntactic style of superposition-based theorem proving, in SCL(EQ) reasoning below variables does not result in an explosion in the number of possibly inferred clauses but also rather in the derivation of more general clauses, see Example 5.

Example 5 (Rewriting below variable level) Assume a term ordering $\prec_{kbo}$, unique weight 1 and with precedence $d \prec c \prec b \prec a \prec g \prec h \prec f$. Further assume β to be large enough. Assume the clause set N:

$$C_1{:=}f(x) \approx h(b) \vee x \not\approx g(a) \qquad C_2{:=}c \approx d \vee f(g(b)) \not\approx h(b)$$
$$C_3{:=}a \approx b \vee f(g(b)) \approx h(b)$$

Let $\sigma = \{x \to g(a)\}$ be a substitution. $C_1\sigma$ must be propagated: $\Gamma = [f(g(a)) \approx h(b)^{0:C_1\sigma}]$. Now suppose that we decide $f(g(b)) \not\approx h(b)$. Then $\Gamma = [f(g(a)) \approx h(b)^{0:C_1\sigma}, f(g(b)) \not\approx h(b)^{1:f(g(b))\not\approx h(b)\vee f(g(b))\approx h(b)}]$ and C_3 is a conflict clause. Explore-

Refutation now creates the following refutation for $a \approx b$:

$$
\begin{aligned}
&I_1{:=}\ (f(x) \approx h(b) \cdot \sigma, C_1 \cdot \sigma, \epsilon, \epsilon, \epsilon)\\
&I_2{:=}\ (f(g(b)) \not\approx h(b), C_2, \epsilon, \epsilon, \epsilon)\\
&I_3{:=}\ (a \approx b, C_3, \epsilon, \epsilon, \epsilon)\\
&I_4{:=}\ (f(g(b)) \approx h(b), f(g(b)) \approx h(b) \vee g(a) \not\approx g(a) \vee f(g(b)) \approx h(b), I_3, I_1, \epsilon)\\
&I_5{:=}\ (h(b) \not\approx h(b), h(b) \not\approx h(b) \vee g(a) \not\approx g(a) \vee f(g(b)) \approx h(b)\\
&\qquad \vee f(g(b)) \approx h(b), I_4, I_2, \epsilon)
\end{aligned}
$$

Multiple applications of Equality-Resolution and Factorize result in the final conflict clause $C_4{:=}f(g(b)) \approx h(b)$ with which we can backtrack. The clause set resulting from this new clause is:

$$
\begin{aligned}
&C_1 = f(x) \approx h(b) \vee x \not\approx g(a) \qquad C_2' = c \approx d\\
&C_4 = f(g(b)) \approx h(b)
\end{aligned}
$$

where C_2' is the result of a unit reduction between C_4 and C_2. Note that the refutation required rewriting below variable level in step I_4. Superposition would create the following clauses (Equality-Resolution and Factorization steps are implicitly done):

$$
\begin{aligned}
N &\Rightarrow_{Sup(C_2,C_3)} N_1 \cup \{C_4{:=}c \approx d \vee a \approx b\}\\
&\Rightarrow_{Sup(C_1,C_2)} N_2 \cup \{C_5{:=}c \approx d \vee g(a) \not\approx g(b)\}\\
&\Rightarrow_{Sup(C_4,C_5)} N_3 \cup \{C_6{:=}c \approx d\}
\end{aligned}
$$

For superposition the resulting clause set is thus:

$$
\begin{aligned}
&C_1 = f(x) \approx h(b) \vee x \not\approx g(a) \qquad C_2 = a \approx b \vee f(g(b)) \approx h(b)\\
&C_6 = c \approx d
\end{aligned}
$$

Currently, the reasoning with solely positive equations is done on and with respect to the trail. It is well-known that also inferences from this type of reasoning can be used to speed up the overall reasoning process. The SCL(EQ) calculus already provides all information for such a type of reasoning, because it computes the justification clauses for trail reasoning via rewriting inferences. By an assessment of the quality of these clauses, e.g., their reduction potential with respect to trail literals, they could also be added, independently from resolving a conflict.

The trail reasoning is currently defined with respect to rewriting. It could also be performed by congruence closure [29]. However, we still need a ground term rewrite system to propagate literals. A possible solution to this is an algorithm by Gallier et al. [22] which creates a ground rewrite system out of the congruence classes in polynomial time.

Bromberger et al. [16] showed how to lift the two-watched literal scheme to SCL. We could make use of this in an implementation as well. In general, an implementation of SCL(EQ) requires both the infrastructure of a superpositon-based prover and an CDCL/SMT solver. The aspect of how to find interesting ground decision or propagation literals for the trail including the respective grounding substituion σ can be treated similar to CDCL [11, 12, 28, 33]. A simple heuristic may be used from the start, like counting the number of instance relationships of some ground literal with respect to the clause set, but later on a bonus system can focus the search towards the structure of the clause sets. Ground literals involved in a conflict or the process of learning a new clause get a bonus or preference. However, since the number of ground literals is not fixed from the beginning with growing β, all these operations need to be done via hashing or indexing operations modulo matching/unification in contrast to simple look-ups in the CDCL case. The regular strategy requires the propagation of all ground unit clauses smaller than β. For an implementation a propagation of the (explicit and

implicit) unit clauses with variables to the trail will be a better choice. This complicates the implementation of refutation proofs and rewriting (congruence closure), but because every reasoning is layered by a ground term β this can still be efficiently done.

Acknowledgements This work was partly funded by DFG Grant 389792660 as part of TRR 248, see https://perspicuous-computing.science. The authors thank the anonymous reviewers and Martin Desharnais for their thorough reading, detailed comments, and corrections.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix: A Proofs and Auxiliary Lemmas

Proof of Lemma 1 Let Γ_1 be a trail and K a defined literal that is of level i in Γ_1. Then K is of level i in a trail $\Gamma{:=}\Gamma_1, \Gamma_2$.

Proof Assume a trail Γ_1 and a literal K that is of level i in Γ_1. Let $\Gamma{:=}\Gamma_1, \Gamma_2$ be a trail. Then we have two cases:

1. K has no defining literal in Γ_1. Then $cores(\Gamma_1; K) = \{[]\}$ contains only the empty core and K is of level 0 in Γ_1. Then $cores(\Gamma; K) = \{[]\}$ as well and thus K is of level 0 in Γ.
2. K has a defining literal $L{:=}max_{\Gamma_1}(K)$ and L is of level i. Then there exists a core $\Delta \in cores(\Gamma_1; K)$ such that L is the maximum literal in Δ according to $\prec_\Gamma$ and for all $\Lambda \in cores(\Gamma_1; K)$ it holds $max_{\prec_\Gamma}(\Delta) \preceq_\Gamma max_{\prec_\Gamma}(\Lambda)$. Thus Δ is a defining core. Now any $\Lambda \in (cores(\Gamma; K) \backslash cores(\Gamma_1; K))$ has a higher maximum literal according to $\prec_\Gamma$. Thus Δ is also a defining core in Γ and L is the defining literal of K in Γ and thus K is of level i in Γ.

□

Auxiliary Lemmas for the Proofs of Lemma 2

Lemma 13 *Let Γ be a trail. Then any literal in Γ occurs exactly once.*

Proof Let $\Gamma{:=}[L_1^{i_1:C_1\cdot\sigma_1}, ..., L_n^{i_n:C_n\cdot\sigma_n}]$. Now suppose there exist L_i, L_j with $i < j$ and $1 \leq i, j \leq n$ such that $L_i = L_j$. By definition of Γ, L_j is undefined in $[L_1, ..., L_i, ..., L_{j-1}]$. But obviously L_j is defined in Γ. Contradiction. □

Lemma 14 *Let Γ be a trail. If a literal L is of level i, then it is not of level $j \neq i$.*

Proof Let Γ be a trail. By Lemma 13 any literal in Γ is unique. Suppose there exists a literal L such that L is of level i and of level j. If the core is empty for L then L is of level 0 by definition. Otherwise there must exist cores $\Delta, \Lambda \in cores(\Gamma; L)$ such that $max_{\prec_\Gamma}(\Delta) \preceq_\Gamma max_{\prec_\Gamma}(\Lambda')$ and $max_{\prec_\Gamma}(\Lambda) \preceq_\Gamma max_{\prec_\Gamma}(\Lambda')$ for all $\Lambda' \in cores(\Gamma; L)$. But then $max_{\prec_\Gamma}(\Lambda) = max_{\prec_\Gamma}(\Delta)$. Contradiction. □

Lemma 15 *Let L be a ground literal and Γ a trail. If L is defined in Γ then L has a level.*

Proof Let Γ be a trail. Suppose that L is defined in Γ. Then it either has a defining literal or it has no defining literal. If it has a defining literal K, then the level of K is the level of L. Since $K \in \Gamma$ it is annotated by a level. Thus L has a level. If L does not have a defining literal, then L is of level 0 by definition of a literal level. □

Proof of Lemma 2-1 $\prec_{\Gamma^*}$ is well-defined.

Proof Suppose a trail $\Gamma := [L_1^{i_1:C_1\cdot\sigma_1}, ..., L_n^{i_n:C_n\cdot\sigma_n}]$ and a term β such that $\{L_1, ..., L_n\} \prec_T \beta$. We have to show that the rules 8.1–8.11 are pairwise disjunct. Consider the rules 8.1–8.7. These rules are pairwise disjunct, if the sets $\{L_1, ..., L_n\}$, $\{comp(L_1), ..., comp(L_n)\}$ and $\{M_{i,j} \mid i \leq n\}$ are pairwise disjunct. Obviously, $\{L_1, ..., L_n\} \cap \{comp(L_1), ..., comp(L_n)\} = \emptyset$. Furthermore $(\{L_1, ..., L_n\} \cup \{comp(L_1), ..., comp(L_n)\}) \cap \{M_{i,j} \mid i \leq n\} = \emptyset$ follows directly from the definition of a trail induced ordering. 8.8 and 8.9 are disjunct since $\{L \mid L\ is\ of\ level\ 0\}$ and $\{L \mid L\ is\ of\ level\ greater\ 0\}$ are disjunct by Lemma 14. It follows that 8.1–8.9 are pairwise disjunct, since all relations in 8.1–8.7 contain only *β-defined* literals of level 1 or higher and all relations in 8.8, 8.9 contain at least one *β-defined* literal of level 0. 8.10 and 8.11 are disjunct since a literal cannot be both *β-defined* and *β-undefined*. It follows that 8.1–8.11 are pairwise disjunct, since all relations in 8.1–8.9 contain only *β-defined* literals and all relations in 8.10, 8.11 contain at least one *β-undefined* literal. □

Proof of Lemma 2-2 $\prec_{\Gamma^*}$ is a total strict order, i.e. $\prec_{\Gamma^*}$ is irreflexive, transitive and total.

Proof Suppose a trail $\Gamma := [L_1^{i_1:C_1\cdot\sigma_1}, ..., L_n^{i_n:C_n\cdot\sigma_n}]$ and a term β such that $\{L_1, ..., L_n\} \prec_T \beta$.

Irreflexivity We have to show that there is no ground literal L such that $L \prec_{\Gamma^*} L$. Suppose two literals L and K such that $L \prec_{\Gamma^*} K$ and $L = K$. Now we have several cases:

1. Suppose that L, K are *β-defined* and of level 1 or higher. Then we have several cases:
   (a) $L = M_{i,j}$ and $K = M_{k,l}$. Then by 8.1 $M_{i,j} \prec_{\Gamma^*} M_{k,l}$ if $i < k$ or ($i = k$ and $j < l$). Thus $i \neq k$ or $j \neq l$. We show that for $M_{i,j}$, $M_{k,l}$ with $i \neq k$ or $j \neq l$ it holds $M_{i,j} \neq M_{k,l}$. Assume that $M_{i,j} = M_{k,l}$ and $k \neq i$ or $j \neq l$. Assume that $k = i$. Then, by Definition 8 $M_{i,j} \prec_T M_{k,l}$ or $M_{k,l} \prec_T M_{i,j}$. Thus $M_{i,j} \neq M_{k,l}$ since $\prec_T$ is a rewrite ordering. Now assume that $k \neq i$. Since $M_{i,j} = M_{k,l}$ it holds $max_\Gamma(M_{i,j}) = max_\Gamma(M_{k,l})$, since both have the same level by Lemma 14. But then $k = i$. Thus $M_{i,j} \neq M_{k,l}$ for $k \neq i$ or $j \neq l$. Thus if by 8.1 $M_{i,j} \prec_{\Gamma^*} M_{k,l}$ if $i < k$ or ($i = k$ and $j < l$), then $M_{i,j} \neq M_{k,l}$.
   (b) $L = L_i$ and $K = L_j$. Then by 8.2 $L_i \prec_{\Gamma^*} L_j$ if $L_i \prec_\Gamma L_j$. Then by Lemma 13 $L_i \neq L_j$.
   (c) $L = comp(L_i)$ and $K = L_j$. Then by 8.3 $comp(L_i) \prec_{\Gamma^*} L_j$ if $L_i \prec_\Gamma L_j$. $L_i \neq L_j$ by Lemma 13. $L \neq K$ has to hold since Γ is consistent.
   (d) $L = L_i$ and $K = comp(L_j)$. Then by 8.4 $L_i \prec_{\Gamma^*} comp(L_j)$ if $L_i \prec_\Gamma L_j$ or $i = j$. If $i \neq j$ then we can proceed analogous to the previous step. If $i = j$ then obviously $L_i \neq comp(L_i)$.
   (e) $L = comp(L_i)$ and $K = comp(L_j)$. Then by 8.5 $comp(L_i) \prec_{\Gamma^*} comp(L_j)$ if $L_i \prec_\Gamma L_j$. By Lemma 13 $L_i \neq L_j$. Thus $comp(L_i) \neq comp(L_j)$.
   (f) $L = L_i$ and $K = M_{k,l}$. Then by 8.6 $L_i \prec_{\Gamma^*} M_{k,l}$, $\text{comp}(L_i) \prec_{\Gamma^*} M_{k,l}$ if $i \leq k$. $M_{k,l} \neq L_i$ and $M_{k,l} \neq comp(L_i)$ follows directly from the Definition 8. Thus if $L_i \prec_{\Gamma^*} M_{k,l}$ or $\text{comp}(L_i) \prec_{\Gamma^*} M_{k,l}$ if $i \leq k$ by 8.6, then $L_i \neq M_{k,l}$ and $\text{comp}(L_i) \neq M_{k,l}$.

(g) $L = M_{k,l}$ and $K = L_i$. Then we can proceed analogous to the previous step for 8.7.

2. Suppose that L and K are *β-defined* and of level zero. Since $\prec_T$ is irreflexive, $L \not\prec_T K$ has to hold. Since $\prec_{\Gamma^*} = \prec_T$ for literals of level zero $L \not\prec_{\Gamma^*} K$ has to hold too.
3. Suppose that L, K are *β-defined* and L is of level zero and K is of level greater than zero. But then $L \neq K$ has to hold by Lemma 14. Thus $L \not\prec_{\Gamma^*} K$ for 8.9.
4. Suppose that L and K are *β-undefined*. Then by 8.10 $K \prec_{\Gamma^*} H$ if $K \prec_T H$. Since $\prec_T$ is a rewrite ordering $K \prec_T H$ iff $K \neq H$.
5. Suppose that L is *β-defined* and K is *β-undefined*. Then by 8.11 $L \prec_{\Gamma^*} K$. Then $L \neq K$ has to hold since otherwise L, K would be both *β-defined* and *β-undefined*, contradicting consistency of Γ.

Transitivity Suppose there exist literals L, K, H such that $H \prec_{\Gamma^*} K$ and $K \prec_{\Gamma^*} L$ but not $H \prec_{\Gamma^*} L$. We have several cases:

1. Suppose all literals are *β-undefined*. Then $K \prec_T L$ and $H \prec_T K$. Otherwise $K \prec_{\Gamma^*} L$ and $H \prec_{\Gamma^*} K$ would not hold. Thus also $H \prec_T L$ by transitivity of $\prec_T$. Thus $H \prec_{\Gamma^*} L$ by 8.10.
2. Suppose two literals are *β-undefined*. If K would be *β-defined*, then $K \prec_{\Gamma^*} H$ by 8.11 contradicting assumption. If L would be *β-defined*, then $L \prec_{\Gamma^*} K$ by 8.11 again contradicting assumption. Thus H has to be *β-defined*. Then $H \prec_{\Gamma^*} L$ by definition 8.11.
3. Suppose one literal is *β-undefined*. If K would be *β-undefined*, then $L \prec_{\Gamma^*} K$ by Definition 8.11 contradicting assumption. If H would be *β-undefined*, then $K \prec_{\Gamma^*} H$ by Definition 8.11 again contradicting assumption. Thus L has to be *β-undefined*. Then $H \prec_{\Gamma^*} L$ by Definition 8.11.
4. Suppose all literals are *β-defined*. Then we have multiple subcases:

   (a) Suppose all literals have the same defining literal L_i and L_i is of level 1 or higher. By 8.6 $L_i \prec_{\Gamma^*} M_{i,j}$ and $comp(L_i) \prec_{\Gamma^*} M_{i,j}$ for all j. By 8.4 $L_i \prec_{\Gamma^*} comp(L_i)$. Thus $L_i \prec_{\Gamma^*} comp(L_i) \prec_{\Gamma^*} M_{i,j}$ for all j. Since $K \prec_{\Gamma^*} L$ either $K = L_i$ and $L \neq L_i$ or $L = M_{i,j}$ and $K = comp(L_i)$ or $L = M_{i,j}$ and $K = M_{i,k}$ with $k < j$.
      (i) Assume $K = L_i$ and $L \neq L_i$. Since K is the smallest literal with defining literal L_i, $K = H$ has to hold. But then $K \prec_{\Gamma^*} K$ contradicting irreflexivity.
      (ii) Assume $L = M_{i,j}$ and $K = comp(L_i)$. Since $H \prec_{\Gamma^*} K$ and all literals have the same defining literal, $H = L_i$ has to hold by 8.6 and 8.4. Then, again by 8.6, $H \prec_{\Gamma^*} L$.
      (iii) $L = M_{i,j}$ and $K = M_{i,k}$ with $k < j$. Since $H \prec_{\Gamma^*} K$ and all have the same defining literal either $H = M_{i,l}$ with $l < k$ by 8.1, or $H = L_i$ or $H = comp(L_i)$ by 8.6. In both cases $H \prec_{\Gamma^*} L$ holds by 8.1 and 8.6.

   (b) Suppose H, K, L have at least one different defining literal and $max_\Gamma(L) = L_i$ with L_i of level 1 or higher. First, we have to show that if $L_j = max_\Gamma(K') \prec_\Gamma max_\Gamma(L') = L_i$ and L_i is of level 1 or higher, then $K' \prec_{\Gamma^*} L'$. Suppose that L_j is of level 0. Then $K' \prec_{\Gamma^*} L'$ by 8.9. Suppose that $L' = M_{i,k}$ and $K' = M_{j,l}$. Then $K' \prec_{\Gamma^*} L'$ by 8.1. Suppose that $L' = M_{i,k}$ and $K' = L_j$ or $K' = comp(L_j)$. Then $K' \prec_{\Gamma^*} L'$ by 8.6. Suppose that $L' = L_i$ or $L' = comp(L_i)$ and $K' = L_j$ or $K' = comp(L_j)$. Then $K' \prec_{\Gamma^*} L'$ by 8.2–8.5. Suppose that $L' = L_i$ or $L' = comp(L_i)$ and $K' = M_{j,l}$. Then $K' \prec_{\Gamma^*} L'$ by 8.7.
   Now by assumption $H \prec_{\Gamma^*} K$ and $K \prec_{\Gamma^*} L$. If $max_\Gamma(K) \prec_\Gamma max_\Gamma(H)$ then $K \prec_{\Gamma^*} H$ contradicting assumption. The same holds for L and K. Thus either $max_\Gamma(H) \prec_\Gamma max_\Gamma(L)$ or $max_\Gamma(K) \prec_\Gamma max_\Gamma(L)$. In the first case $H \prec_{\Gamma^*} L$

follows from above. In the second case $max_\Gamma(H) \preceq_\Gamma max_\Gamma(K)$ has to hold. Thus $H \prec_{\Gamma^*} L$ follows again.

(c) Suppose that $max_\Gamma(L) = L_i$ where L_i is of level 0. Since $K \prec_{\Gamma^*} L, max_\Gamma(K) = L_j$ with L_j of level 0 has to hold by 8.9 and 8.11. Now assume that $L \prec_T K$. Then $L \prec_{\Gamma^*} K$ by 8.8 contradicting assumption. Thus $K \prec_T L$ has to hold since $K \neq L$. Since $H \prec_{\Gamma^*} K$, $max_\Gamma(H) = L_k$ with L_k of level 0 has to hold by 8.9 and 8.11. Now assume that $K \prec_T H$. Then $K \prec_{\Gamma^*} H$ by 8.8 contradicting assumption. Thus $H \prec_T K$ has to hold since $H \neq K$. By transitivity of $\prec_T$, $H \prec_T L$ and thus $H \prec_{\Gamma^*} L$ has to hold.

Totality First we show that any ground literal is either *β-defined* and has a level or *β-undefined*. Since Γ is consistent, a literal is either *β-defined* or *β-undefined*. We just need to show that if a literal is *β-defined*, it has a level. By Lemma 15 all defined literals have a level. *β-definedness* implies definedness. Thus all *β-defined* literals have a level. Now assume some arbitrary ground literals $L \neq K$. We have several cases:

1. L, K are *β-undefined*. Since $L \neq K$ we have $L \prec_T K$ or $K \prec_T L$ by totality of $\prec_T$ on ground literals. Thus by 8.10 $L \prec_{\Gamma^*} K$ or $K \prec_{\Gamma^*} L$.
2. One is *β-defined*. Then either $L \prec_{\Gamma^*} K$ or $K \prec_{\Gamma^*} L$ by 8.11.
3. Both are *β-defined*. Then we have several subcases:

(a) L is of level zero and K is of level greater than zero or vice versa. Then by 8.9 $L \prec_{\Gamma^*} K$ or $K \prec_{\Gamma^*} L$ has to hold.

(b) $max_\Gamma(L) = L_i$ and $max_\Gamma(K) = L_j$ and both are of level 1 or higher.

(i) $L = M_{i,k}$ and $K = M_{j,l}$. Then either $M_{i,k} \prec_{\Gamma^*} M_{j,l}$ or $M_{j,l} \prec_{\Gamma^*} M_{i,k}$ by 8.1.

(ii) $L = M_{i,k}$ and $K = L_j$ or $K = comp(L_j)$. If $i \geq j$ then by 8.6 $L_j \prec_{\Gamma^*} M_{i,k}$ or $comp(L_j) \prec_{\Gamma^*} M_{i,k}$. If $i < j$ then by 8.7 $M_{i,k} \prec_{\Gamma^*} L_j$ or $M_{i,k} \prec_{\Gamma^*} comp(L_j)$.

(iii) $K = M_{i,k}$ and $L = L_j$ or $L = comp(L_j)$. Analogous to previous step.

(iv) $L = L_i$ and $K = L_j$. Then if $i < j$ by 8.2 $L_i \prec_{\Gamma^*} L_j$ and $L_j \prec_{\Gamma^*} L_i$ otherwise.

(v) $L = comp(L_i)$ and $K = comp(L_j)$ analogous to previous step for 8.5.

(vi) $L = L_i$ and $K = comp(L_j)$. Then if $i \leq j$ by 8.4 $L_i \prec_{\Gamma^*} comp(L_j)$. If $j < i$ by 8.3 $comp(L_j) \prec_{\Gamma^*} L_i$.

(vii) $L = comp(L_i)$ and $K = L_j$. Then if $i < j$ by 8.3 $comp(L_i) \prec_{\Gamma^*} L_j$. If $j \leq i$ by 8.4 $L_j \prec_{\Gamma^*} comp(L_i)$.

(c) $max_\Gamma(L) = L_i$ and $max_\Gamma(K) = L_j$ and both are of level 0. Now either $L \prec_T K$ or $K \prec_T L$. Thus by 8.8 $L \prec_{\Gamma^*} K$ or $K \prec_{\Gamma^*} L$.

□

Proof of Lemma 2-3 $\prec_{\Gamma^*}$ is a well-founded ordering.

Proof Suppose some arbitrary subset M of all ground literals, a trail $\Gamma := [L_1^{i_1:C_1\cdot\sigma_1}, ..., L_n^{i_n:C_n\cdot\sigma_n}]$ and a term β such that $\{L_1, ..., L_n\} \prec_T \beta$. We have to show that M has a minimal element. We have several cases:

1. L is *β-undefined* in Γ for all literals $L \in M$. Then $\prec_{\Gamma^*} = \prec_T$. Since $\prec_T$ is well-founded there exists a minimal element in M. Thus there exists a minimal element in M with regard to $\prec_{\Gamma^*}$.
2. there exists at least one literal in M that is *β-defined*. Then we have two cases:

(a) there exists a literal in M that is of level zero. Then let $L \in M$ be the literal of level zero, where $L \prec_T K$ for all $K \in M$ with K of level zero. We show that L is the

minimal element. Suppose there exists a literal $L' \in M$ that is smaller than L. Since L is of level zero, $L \prec_{\Gamma^*} K$ for all literals K of level greater than zero by 8.9 and $L \prec_{\Gamma^*} H$ for all β-*undefined* literals H by 8.11. Thus L' must be of level zero. But then $L' \prec_T L$ has to hold, contradicting assumption.

(b) There exists no literal in M that is of level zero. Let $L \in M$ be the literal where $max_\Gamma(L) \preceq_\Gamma max_\Gamma(K)$ for all $K \in M$ and
   (i) $L = max_\Gamma(L)$ or;
   (ii) $L = comp(max_\Gamma(L))$ and $max_\Gamma(L) \notin M$ or;
   (iii) $L \prec_T H$ for all $H \in M$ such that $max_\Gamma(L) = max_\Gamma(H)$ and $max_\Gamma(L) \notin M$ and $comp(max_\Gamma(L)) \notin M$.

   We show that L is the minimal element. Suppose there exists a literal $L' \in M$ that is smaller than L. We have three cases:
   (i) $max_\Gamma(L) = L = L_i$. Since $L_i \prec_T \beta$ we have either $L' = L_j$ with $j < i$ by 8.2 or $L' = comp(L_j)$ with $j < i$ by 8.3 or $L' = M_{k,l}$ with $k < i$ by 8.7. In all three cases we have $max_\Gamma(L') \prec_\Gamma max_\Gamma(L)$ contradicting assumption that the defining literal of L is minimal in M.
   (ii) $L = comp(L_i) = comp(max_\Gamma(L))$ and $max_\Gamma(L) \notin M$. Since $L_i \prec_T \beta$ either $L' = L_j$ with $j < i$ by 8.4 or $L' = comp(L_j)$ with $j < i$ by 8.5 or $L' = M_{k,l}$ with $k < i$ by 8.7. In all three cases we have $max_\Gamma(L') \prec_\Gamma max_\Gamma(L)$ contradicting assumption that the defining literal of L is minimal in M.
   (iii) $L = M_{k,l}$ and $max_\Gamma(L) \notin M$ and $comp(max_\Gamma(L)) \notin M$. Then either $L' = M_{i,j}$ with $i < k$ or ($i = k$ and $j < l$) by 8.1 or $L' = L_i$ or $L' = comp(L_i)$ with $i \leq k$. Suppose that $L' = M_{i,j}$ and $i < k$. Then $max_\Gamma(L') \prec_\Gamma max_\Gamma(L)$ contradicting assumption. Suppose that $L' = M_{i,j}$ and $i = k$ and $j < l$. Then $L' \prec_T L$ and $max_\Gamma(L) = max_\Gamma(L')$. For L it holds $L \prec_T H$ for all $H \in M$ such that $max_\Gamma(L) = max_\Gamma(H)$. Contradiction. Suppose that $L' = L_i$ or $L' = comp(L_i)$ with $i = k$. Then $max_\Gamma(L) = L_i$. By assumption $max_\Gamma(L) \notin M$ and $comp(max_\Gamma(L)) \notin M$. Contradiction. Suppose that $L' = L_i$ or $L' = comp(L_i)$ with $i < k$. Then we have $max_\Gamma(L') \prec_\Gamma max_\Gamma(L)$ contradicting assumption that the defining literal of L is minimal in M. □

Proof of Lemma 4 Assume that all ground terms t with $t \prec_T \beta$ for any β are polynomial in the size of β. Then testing Propagate (Conflict) is NP-Complete, i.e., the problem of checking for a given clause C whether there exists a grounding substitution σ such that $C\sigma$ propagates (is false) is NP-Complete.

Proof Let $C\sigma$ be propagable (false). The problem is in NP because β is constant and for all $t \in codom(\sigma)$ it holds that t is polynomial in the size of β. Checking if $C\sigma$ is propagable (false) can be done in polynomial time with Congruence Closure [29] since σ has polynomial size.

We reduce 3-SAT to testing rule Conflict. Consider a 3-place predicate R, a unary function g, and a mapping from propositional variables P to first-order variables x_P. Assume a 3-SAT clause set $N = \{\{L_0, L_1, L_2\}, ..., \{L_{n-2}, L_{n-1}, L_n\}\}$, where L_i may denote both P_i and $\neg P_i$. Now we create the clause

$$\{R(t_0, t_1, t_2) \not\approx true, ..., R(t_{n-2}, t_{n-1}, t_n \not\approx true)\},$$

where $t_i := x_{P_i}$ if $L_i = P_i$ and $t_i := g(x_{P_i})$ otherwise. Now let $\Gamma := \{R(x_0, x_1, x_2) \mid x_i \in \{0, 1, g(0), g(1)\}$ such that $(x_0 \vee x_1 \vee x_2) \downarrow_{\{g(x) \mapsto (\neg x)\}}$ is true $\}$ be the set of all R-atoms that evaluate to true if considered as a three literal propositional clause. Now N is satisfiable if

and only if Conflict is applicable to the new clause. The reduction is analogous for Propagate. □

Proof of Theorem 1 Assume a state $(\Gamma; N; U; \beta; k; D)$ resulting from a run. Then $(\Gamma; N; U; \beta; k; D)$ is sound.

Proof Proof by structural induction on $(\Gamma; N; U; \beta; k; D)$. Let $(\Gamma; N; U; \beta; k; D) = (\epsilon, N, \emptyset, \beta, 0, \top)$, the initial state. Then it is sound according to Lemma 5. Now assume that $(\Gamma; N; U; \beta; k; D)$ is sound. We need to show that any application of a rule results in a sound state.

Propagate Assume *Propagate* is applicable. Then there exists $C \in N \cup U$ such that $C = C_0 \vee C_1 \vee L$, $L\sigma$ is *β-undefined* in Γ, $C_1\sigma = L\sigma \vee ... \vee L\sigma$, $C_1 = L_1 \vee ... \vee L_n$,$\mu = mgu(L_1, ..., L_n, L)$ and $C_0\sigma$ is *β-false* in Γ. Then a reduction chain application $[I_1, ..., I_m]$ from Γ to $L\sigma^{k:(C_0 \vee L)\mu\cdot\sigma}$ is created with $I_m{:=}(s_m\#t_m \cdot \sigma_m, s_m\#t_m \vee C_m \cdot \sigma_m, I_j, I_k, p_m)$. Finally $s_m\#t_m\sigma_m^{k:(s_m\#t_m \vee C_m)\cdot\sigma_m}$ is added to Γ.

By definition of a reduction chain application $(s_m\#t_m)\sigma_m = L\sigma\downarrow_{conv(\Gamma)}$. Thus, $(s_m\#t_m)\sigma_m$ must be *β-undefined* in Γ and irreducible by $conv(\Gamma)$, since $(C_0 \vee L)\mu\sigma \prec_T \beta$ by definition of Propagate.

- 13.1: Since $(s_m\#t_m)\sigma_m$ is *β-undefined* in Γ, adding $(s_m\#t_m)\sigma_m$ does not make Γ inconsistent. Thus Γ, $(s_m\#t_m)\sigma_m$ remains consistent.
- 13.2: $(s_m\#t_m)\sigma_m$ is *β-undefined* in Γ and irreducible by $conv(\Gamma)$. It remains to show that $C_m\sigma_m$ is *β-false* in Γ, $N \cup U \models s_m\#t_m \vee C_m$ and $(s_m\#t_m \vee C_m)\sigma_m \prec_T \beta$. By i.h. for all $L'\sigma'^{l:(L' \vee C')\cdot\sigma'} \in \Gamma$ it holds that $C'\sigma'$ is *β-false* in Γ, $(L' \vee C')\sigma' \prec_T \beta$ and $N \cup U \models (L' \vee C')$. By definition of Propagate $C_0\sigma$ is *β-false* in Γ and $C\sigma \prec_T \beta$ and $N \cup U \models C$. $(C_0 \vee C_1 \vee L)\mu$ is an instance of C. Thus $C \models (C_0 \vee C_1 \vee L)\mu$. $C_0\mu = L\mu \vee ... \vee L\mu$ by definition of Propagate. Thus $C \models (C_1 \vee L)\mu$ and by this $N \cup U \models (C_1 \vee L)\mu$. By definition of a reduction chain application I_j either contains a clause annotation from Γ, $L\sigma^{k:(C_0 \vee L)\cdot\sigma}$ or it is a rewriting inference from smaller rewrite steps for all $1 \leq j \leq m$. Thus, by Lemma 3 it follows by induction that for any rewriting inference $I_j{:=}(s_j\#t_j \cdot \sigma_j, s_j\#t_j \vee C_j \cdot \sigma_j, I_i, I_k, p_j)$ it holds $C_j\sigma_j$ is *β-false* in Γ, $N \cup U \models s_j\#t_j \vee C_j$ and $(s_j\#t_j \vee C_j)\sigma_j \prec_T \beta$.
- 13.3 and 13.4: trivially hold by induction hypothesis.
- 13.5: trivially holds since $D = \top$.

Decide Assume *Decide* is applicable. Then there exists $C \in N \cup U$ such that $C = C_0 \vee L$, $L\sigma$ is ground and *β-undefined* in Γ and $C_0\sigma$ is ground and *β-undefined* or *β-true* in Γ. Then a reduction chain application $[I_1, ..., I_m]$ from Γ to $L\sigma^{k+1:(C_0 \vee L)\cdot\sigma}$ is created with $I_m{:=}(s_m\#t_m \cdot \sigma_m, s_m\#t_m \vee C_m \cdot \sigma_m, I_j, I_k, p_m)$. Finally $s_m\#t_m\sigma_m^{k+1:(s_m\#t_m \vee comp(s_m\#t_m))\cdot\sigma_m}$ is added to Γ.

By definition of a reduction chain application $(s_m\#t_m)\sigma_m = L\sigma\downarrow_{conv(\Gamma)}$. Thus, $(s_m\#t_m)\sigma_m$ must be *β-undefined* in Γ and irreducible by $conv(\Gamma)$, since $(C_0 \vee L)\sigma \prec_T \beta$ by definition of Decide.

- 13.1: Since $(s_m \# t_m)\sigma_m$ is *β-undefined* in Γ adding $(s_m\#t_m)\sigma_m$ does not make Γ inconsistent. Thus Γ, $(s_m\#t_m)\sigma_m$ remains consistent.
- 13.3: $(s_m\#t_m)\sigma_m$ is *β-undefined* in Γ and irreducible by $conv(\Gamma)$. $N \cup U \models (s_m\#t_m) \vee comp(s_m\#t_m)$ obviously holds. $(s_m\#t_m)\sigma_m \prec_T \beta$ holds inductively by Lemma 3 and since $L\sigma \prec_T \beta$.
- 13.2 and 13.4 trivially hold by induction hypothesis.
- 13.5: trivially holds since $D = \top$.

Conflict Assume *Conflict* is applicable. Then there exists a $D'\sigma$ such that $D'\sigma$ is β-*false* in Γ. Then:

- 13.1–13.4: trivially hold by induction hypothesis
- 13.5: $D'\sigma$ is β-*false* in Γ by definition of *Conflict*. Now we have two cases:
  1. $D'\sigma$ is of level greater than zero. Then $N \cup U \models D'$ since $D' \in N \cup U$ by definition of *Conflict*.
  2. $D'\sigma$ is of level zero. Then we have to show that $N \cup U \models \bot$. For any literal $L_0^{0:(L_0 \vee D_0)\cdot\sigma} \in \Gamma$ it holds $N \models L_0$, since any literal of level 0 is a propagated literal. By definition of a level, for any $K \in D'\sigma$ there exists a core $core(\Gamma; K)$ that contains only literals of level 0. Thus $N \cup U \models core(\Gamma; K)$ and $core(\Gamma; K) \models \neg K$ for any such K. Then $N \cup U \models \neg D'\sigma$ and $N \cup U \models D'\sigma$ and therefore $N \cup U \models \bot$.

Skip Assume *Skip* is applicable. Then $\Gamma = \Gamma', L$ and $D = D' \cdot \sigma$ and $D'\sigma$ is β-*false* in Γ'.

- 13.1: By i.h. Γ is consistent. Thus Γ' is consistent as well.
- 13.2–13.4: trivially hold by induction hypothesis and since Γ' is a prefix of Γ.
- 13.5: By i.h. $D'\sigma$ is β-*false* in Γ and $N \cup U \models D'$. By definition of *Skip* $D'\sigma$ is β-*false* in Γ'.

Explore-Refutation Assume *Explore-Refutation* is applicable. Then $D = (D' \vee s \,\#\, t) \cdot \sigma$, $(s \,\#\, t)\sigma$ is strictly $\prec_{\Gamma^*}$ maximal in $(D' \vee s \,\#\, t)\sigma$, $[I_1, ..., I_m]$ is a refutation from Γ and $(s \,\#\, t)\sigma$, $I_j = (s_j \# t_j \cdot \sigma_j, (s_j \# t_j \vee C_j) \cdot \sigma_j, I_l, I_k, p_j), 1 \leq j \leq m, (s_j \,\#\, t_j \vee C_j)\sigma_j \prec_{\Gamma^*} (D' \vee s \,\#\, t)\sigma$, $(s_j \# t_j \vee C_j)\sigma_j$ is β-*false* in Γ.

- 13.1–13:4 trivially hold by i.h.
- 13.5: By definition $(C_j \vee s_j \,\#\, t_j)\sigma_j$ is β-*false* in Γ. By i.h. for all $L'\sigma'^{l:(L' \vee C')\cdot\sigma'} \in \Gamma$ it holds that $N \cup U \models (L' \vee C')$. By i.h. $N \cup U \models D' \vee s \,\#\, t$. By definition of a refutation $I_j := (s_j \# t_j \cdot \sigma_j, s_j \# t_j \vee C_j \cdot \sigma_j, I_i, I_k, p_j)$ either contains a clause annotation from Γ, $(s \,\#\, t)\sigma^{k:(D' \vee s \,\#\, t)\cdot\sigma}$ or it is a rewriting inference from smaller rewrite steps for all $1 \leq j \leq m$. Thus it follows inductively by Lemma 3 that $N \cup U \models (s_j \# t_j \vee C_j)$.

Factorize Assume *Factorize* is applicable. Then $D = D' \cdot \sigma$.

- 13.1–13.4: trivially hold by induction hypothesis.
- 13.5: By i.h. $D'\sigma$ is β-*false* in Γ and $N \cup U \models D'$. By the definition of *Factorize* $D' = D_0 \vee L \vee L'$ such that $L\sigma = L'\sigma$ and $\mu = mgu(L, L')$. $(D_0 \vee L \vee L')\mu$ is an instance of D'. Thus $N \cup U \models (D_0 \vee L \vee L')\mu$. Since $L\mu = L'\mu$, $(D_0 \vee L \vee L')\mu \models (D_0 \vee L)\mu$. Thus $N \cup U \models (D_0 \vee L)\mu$ and $(D_0 \vee L)\mu\sigma$ is β-*false* since $(D_0 \vee L)\mu\sigma = (D_0 \vee L)\sigma$ by definition of an mgu.

Equality-Resolution Assume *Equality-Resolution* is applicable. Then $D = (D' \vee s \not\approx s')\sigma$ and $s\sigma = s'\sigma$, $\mu = mgu(s, s')$. Then

- 13.1–13.4: trivially hold by induction hypothesis.
- 13.5: By i.h. $(D' \vee s \not\approx s')\sigma$ is β-*false* in Γ and $N \cup U \models (D' \vee s \not\approx s')$. $D'\mu$ is an instance of $(D' \vee s \not\approx s')$. Thus $(D' \vee s \not\approx s') \models D'\mu$. Thus $N \cup U \models D'\mu$. $D'\mu\sigma$ is β-*false* since $(D' \vee s \not\approx s')\sigma$ is β-*false* and $D'\mu\sigma = D'\sigma$ by definition of a mgu.

Backtrack Assume *Backtrack* is applicable. Then $\Gamma = \Gamma', K, \Gamma''$ and $D = (D' \vee L)\sigma$, where $L\sigma$ is of level k, and $D'\sigma$ is of level i.

- 13.1: By i.h. Γ is consistent. Thus $\Gamma' \subseteq \Gamma$ is consistent.

- 13.2–13.3: Since Γ' is a prefix of Γ by i.h. this holds.
- 13.4: By i.h. $N \cup U \models D' \vee L$ and $N \models U$. Thus $N \models U \cup \{D' \vee L\}$.
- 13.5: trivially holds since $D = \top$ after backtracking.

□

Proof of Lemma 7 Assume a state $(\Gamma; N; U; \beta; k; D)$ resulting from a regular run where the current level $k > 0$ and a unit clause $l \approx r \in N$. Now assume a clause $C \vee L[l']_p \in N$ such that $l' = l\mu$ for some matcher μ. Now assume some arbitrary grounding substitutions σ' for $C \vee L[l']_p$, σ for $l \approx r$ such that $l\sigma = l'\sigma'$ and $r\sigma \prec_T l\sigma$. Then $(C \vee L[r\mu\sigma\sigma']_p)\sigma' \prec_{\Gamma^*} (C \vee L[l']_p)\sigma'$.

Proof Let $(\Gamma; N; U; \beta; k; D)$ be a state resulting from a regular run where $k > 0$ and $\Gamma = [L_1, ..., L_n]$. Now we have two cases:

1. $\beta \prec_T (l \approx r)\sigma$. Since $(l \approx r)\sigma$ rewrites $L[l']_p\sigma'$, $\beta \prec_T L[l']_p\sigma'$ has to hold as well. Thus $(l \approx r)\sigma$ is *β-undefined* in Γ and $L[l']_p\sigma'$ is *β-undefined* in Γ. By definition of a trail induced ordering $\prec_{\Gamma^*} := \prec_T$ for *β-undefined* literals. Thus, in case that $L[r\mu]_p)\sigma\sigma'$ is still undefined, $(L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (L[l']_p)\sigma'$ has to hold since $(L[r\mu]_p)\sigma\sigma' \prec_T (L[l']_p)\sigma'$. Thus, according to the definition of multiset orderings, $(C \vee L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (C \vee L[l']_p)\sigma'$. In the case that $(L[r\mu]_p)\sigma\sigma'$ is defined, $(L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (L[l']_p)\sigma'$ has to hold as well by Definition 8.11. Thus, according to the definition of multiset orderings, $(C \vee L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (C \vee L[l']_p)\sigma'$.
2. $(l \approx r)\sigma \prec_T \beta$. Since propagation is exhaustive for literals of level 0 (cf. 15.2) $(l \approx r)\sigma$ is on the trail or defined and of level 0. Now we have two cases:
   (a) $(L[l']_p)\sigma'$ is of level 1 or higher. Since $(L[l']_p)\sigma'$ is reducible by $(l \approx r)\sigma$, $(L[l']_p)\sigma' \neq L_i$ and $(L[l']_p)\sigma' \neq comp(L_i)$ for all $L_i \in \Gamma$. Since $(L[l']_p)\sigma'$ is of level 1 or higher, rewriting with $(l \approx r)\sigma$ does not change the defining literal of $(L[l']_p)\sigma'$. Thus $(L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (L[l']_p)\sigma'$ has to hold since $(L[r\mu]_p)\sigma\sigma' \prec_T (L[l']_p)\sigma'$. Thus, according to the definition of multiset orderings, $(C \vee L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (C \vee L[l']_p)\sigma'$.
   (b) $(L[l']_p)\sigma'$ is of level 0. First we show that $(L[r\mu]_p)\sigma\sigma'$ is still of level 0. Suppose that $(L[l']_p)\sigma' = s \# s$. Then rewriting either the left or right side of the equation results in $(L[r\mu]_p)\sigma\sigma'$. Then $core(\Gamma; (l \approx r)\sigma)$ is also a core for $(L[r\mu]_p)\sigma\sigma'$ and thus $(L[r\mu]_p)\sigma\sigma'$ must be of level 0. Now suppose that $(L[r\mu]_p)\sigma\sigma' = s \# s$. Then it is of level 0 by definition of a level. Finally suppose that $(L[r\mu]_p)\sigma\sigma' \neq s \# s$ and $(L[l']_p)\sigma' \neq s \# s$. Then $core(\Gamma; (L[l']_p)\sigma') \cup core(\Gamma; (l \approx r)\sigma)$ is a core for $(L[r\mu]_p)\sigma\sigma'$. Thus $(L[r\mu]_p)\sigma\sigma'$ is of level 0. Since $(L[r\mu]_p)\sigma\sigma' \prec_T (L[l']_p)\sigma'$, $(L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (L[l']_p)\sigma'$ according to the definition of $\prec_{\Gamma^*}$. Thus, according to the definition of multiset orderings, $(C \vee L[r\mu]_p)\sigma\sigma' \prec_{\Gamma^*} (C \vee L[l']_p)\sigma'$.

□

Proof of Lemma 8 Let C, D be two clauses. If there exists a substitution σ such that $C\sigma \subset D$, then D is redundant with respect to C and any $\prec_{\Gamma^*}$.

Proof Let τ be a grounding substitution for D. Since $C\sigma \subset D$, $C\sigma\tau \subset D\tau$. Thus, for any $L \in C\sigma\tau$ it holds $L \in D\tau$ and $C\sigma\tau \neq D\tau$. Thus, $C\sigma\tau \prec_{\Gamma^*} D\tau$ by definition of a multiset extension and $C\sigma\tau$ makes $D\tau$ redundant by Definition 1. □

Auxiliary Lemmas for the Proof of Lemma 6

Lemma 16 *During a regular run, if* $(\Gamma; N; U; \beta; k; \top)$ *is the immediate result of an application of* $Backtrack$*, then there exists no clause* $C \in N \cup U$ *and no substitution* σ *such that* $C\sigma$ *is* β*-false in* Γ.

Proof We prove this by induction. For the induction start assume the state $(\Gamma'; N; U \cup \{D\}; \beta; i; \top)$ after the first application of $Backtrack$ in a regular run, where D is the learned clause. Since $Backtrack$ was not applied before, the previous (first) application of *Conflict* in a state $(\Gamma, K; N; U; \beta; k; \top)$ was immediately preceded by an application of $Propagate$ or $Decide$. By the definition of a regular run there is no clause $C \in N$ with substitution σ such that $C\sigma$ is β*-false* in Γ. Otherwise *Conflict* would have been applied earlier. By the definition of $Backtrack$, there exists no substition τ such that $D\tau$ is β*-false* in Γ'. Since there existed such a substitution before the application of $Backtrack$, Γ' has to be a prefix of Γ and $\Gamma \neq \Gamma'$. Thus there exists no clause $C \in N \cup U \cup \{D\}$ and a grounding substitution δ such that $C\delta$ is β*-false* in Γ'.

For the induction step assume the state $(\Gamma'; N; U \cup \{D\}; \beta; i; \top)$ after nth application of $Backtrack$. By i.h. the previous application of $Backtrack$ did not produce any β*-false* clause. It follows that the the previous application of *Conflict* in a state $(\Gamma, K; N; U; \beta; k; \top)$ was immediately preceded by an application of $Propagate$ or $Decide$. By the definition of a regular run there is no clause $C \in N \cup U$ with substitution σ such that $C\sigma$ is β*-false* in Γ. Otherwise *Conflict* would have been applied earlier. By the definition of $Backtrack$, there exists no substition τ such that $D\tau$ is β*-false* in Γ'. Since there existed such a substitution before the application of $Backtrack$, Γ' has to be a prefix of Γ and $\Gamma \neq \Gamma'$. Thus there exists no clause $C \in N \cup U \cup \{D\}$ and a grounding substitution δ such that $C\delta$ is β*-false* in Γ'. □

Corollary 2 *If Conflict is applied in a regular run, then it is immediately preceded by an application of* $Propagate$ *or* $Decide$*, except if it is applied to the initial state.*

Lemma 17 *Assume a state* $(\Gamma; N; U; \beta; k; D)$ *resulting from a regular run. Then there exists no clause* $(C \vee L) \in N \cup U$ *and no grounding substitution* σ *such that* $(C \vee L)\sigma$ *is* β*-false in* Γ*,* $comp(L\sigma)$ *is a decision literal of level* i *in* Γ *and* $C\sigma$ *is of level* $j < i$.

Proof Proof is by induction. Assume the initial state $(\epsilon; N; \emptyset; \beta; 0; \top)$. Then any clause $C \in N$ is undefined in Γ. Then this trivially holds.

Now for the induction step assume a state $(\Gamma; N; U; \beta; k; D)$. Only $Propagate$, $Decide$, $Backtrack$ and $Skip$ change the trail and only $Backtrack$ adds a new literal to U. By i.h. there exists no clause with the above properties in $N \cup U$.

Now assume that $Propagate$ is applied. Then a literal L is added to the trail. Let $C_1 \vee L_1, ..., C_n \vee L_n$ be the ground clause instances that get β*-false* in Γ by the application such that L is the defining literal of $L_1, ..., L_n$. Then L_i is of level k for $1 \leq i \leq n$. Thus $L_i \neq comp(K)$ for the decision literal $K \in \Gamma$ of level k. Thus $C_1 \vee L_1, ..., C_n \vee L_n$ do not have the above properties.

Now assume that $Decide$ is applied. Then a literal L of level $k+1$ is added to the trail. Let $C_1 \vee L_1, ..., C_n \vee L_n$ be the (ground) clause instances that get β*-false* in Γ by the application such that L is the defining literal of $L_1, ..., L_n$. By the definition of a regular run for all L_i with $1 \leq i \leq n$ it holds that $L_i \neq comp(L)$ or there exists another literal $K_i \in C_i$ such that K_i is of level $k+1$ and $L_i \neq K_i$, since otherwise $Propagate$ must be applied. Thus $C_1 \vee L_1, ..., C_n \vee L_n$ do not have the above properties.

Now assume that $Skip$ is applied. Then there are no new clauses that get β*-false* in Γ. Thus this trivially holds.

Now assume that *Backtrack* is applied. Then a new clause $D \vee L$ is added to U and $\Gamma = \Gamma', K, \Gamma''$ such that there is a grounding substitution τ with $(D \vee L)\tau$ *β-false* in Γ', K, there is no grounding substitution δ with $(D \vee L)\delta$ *β-false* in Γ'. Γ' is the trail resulting from the application of *Backtrack*. By Lemma 16, after application of *Backtrack* there exists no clause $C \in N \cup U$ and a substitution σ such that $C\sigma$ is *β-false* in Γ'. Thus there exists no clause with the above properties. □

Proof of Lemma 6 Let N be a clause set. The clauses learned during a regular run in SCL(EQ) are not redundant with respect to $\prec_{\Gamma^*}$ and $N \cup U$. For the trail only non-redundant clauses need to be considered.

We first prove that learned clauses are non-redundant and then that only non-redundant clauses need to be considered, Lemma 21, below.

Proof Consider the following fragment of a derivation learning a clause:

$$\begin{aligned}
&\Rightarrow^{Conflict}_{\text{SCL(EQ)}} && (\Gamma; N; U; \beta; k; D \cdot \sigma)\\
&\Rightarrow^{\{Explore\text{-}Refutation, Skip, Eq\text{-}Res, Factorize\}^*}_{\text{SCL(EQ)}} && (\Gamma'; N; U; \beta; l; C \cdot \sigma)\\
&\Rightarrow^{Backtrack}_{\text{SCL(EQ)}} && (\Gamma''; N; U \cup \{C\}; \beta; k'; \top).
\end{aligned}$$

Assume there are clauses in $N' \subseteq (gnd(N \cup U)^{\preceq_{\Gamma^*} C\sigma})$ such that $N' \models C\sigma$. Since $N' \preceq_{\Gamma^*} C\sigma$ and $C\sigma$ is *β-defined* in Γ, there is no *β-undefined* literal in N', as all *β-undefined* literals are greater than all *β-defined* literals. If $\Gamma \models N'$ then $\Gamma \models C\sigma$, a contradiction. Thus there is a $C' \in N'$ with $C' \preceq_{\Gamma^*} C\sigma$ such that C' is *β-false* in Γ. Now we have two cases:

1. $\Gamma' \neq \Gamma$. Then $\Gamma = \Gamma', \Delta$. Thus at least one Skip was applied, so $C\sigma$ does not contain a literal that is *β-undefined* without the rightmost literal of Γ, therefore $C\sigma \neq D\sigma$. Suppose that this is not the case, so $C\sigma = D\sigma$. Then $D\sigma$ is *β-false* in Γ'. But since *Backtrack* does not produce any *β-false* clauses by Lemma 16, *Conflict* could have been applied earlier on $D\sigma$ contradicting a regular run. Since $C' \preceq_{\Gamma^*} C\sigma$ we have that $C' \neq D\sigma$ as well. Thus, again since *Backtrack* does not produce any *β-false* clauses by Lemma 16, at a previous point in the derivation there must have been a state such that C' was *β-false* under the current trail and *Conflict* was applicable but not applied, a contradiction to the definition of a regular run.
2. $\Gamma' = \Gamma$, then conflict was applied immediately after an application of *Decide* by Corollary 2 and the definition of a regular run. Thus $\Gamma = \Delta, K^{(k-1):D'\cdot\delta}, L^{k:D''\cdot\tau}$. C' does not have any *β-undefined* literals. Suppose that C' has no literals of level k. Then all literals in C' are of level $i < k$. Since C' is *β-false* in Γ, C' is *β-false* in Δ, K as well, since it does not have any literals of level k. Thus, again since *Backtrack* does not produce any *β-false* clauses by Lemma 16, at a previous point in the derivation there must have been a state such that C' was *β-false* under the current trail and *Conflict* was applicable but not applied, a contradiction to the definition of a regular run.
   Since $C' \preceq_{\Gamma^*} C\sigma$, it may have at most one literal of level k, namely $comp(L)$, since $comp(L) \in C\sigma$ by definition of a regular run, since Skip was not applied, and there exists only L such that $L \prec_{\Gamma^*} comp(L)$ and L is of level k. But L is *β-true* in Γ. Thus $L \notin C'$ has to hold.
   Now suppose that C' has one literal of level k. Thus $C' = C'' \vee comp(L)$, where C'' is *β-false* in Δ, K. But by Lemma 17 there does not exist such a clause. Contradiction.

□

Auxiliary Lemma for the Proof of Lemma 19

Lemma 18 *Assume a clause $L_1 \vee ... \vee L_m$, a trail Γ resulting from a regular run starting from the initial state, and a reducible (by $conv(\Gamma)$) grounding substitution σ, such that $L_i\sigma$ is β-false (β-true or β-undefined) in Γ and $L_i\sigma \prec_T \beta$ for $1 \leq i \leq m$. Then there exists a substitution σ' that is irreducible by $conv(\Gamma)$ such that $L_i\sigma'$ is β-false (β-true or β-undefined) in Γ, $L_i\sigma' \prec_T \beta$ and $L_i\sigma{\downarrow_{conv(\Gamma)}} = L_i\sigma'{\downarrow_{conv(\Gamma)}}$.*

Proof Let $L_1 \vee ... \vee L_m$ be a clause, Γ a trail resulting from a regular run. Let $\sigma := \{x_1 \to t_1, ..., x_n \to t_n\}$. Now set $\sigma' := \{x_1 \to (t_1{\downarrow_{conv(\Gamma)}}), ..., x_n \to (t_n{\downarrow_{conv(\Gamma)}})\}$. Obviously σ' is irreducible by $conv(\Gamma)$ and $L_i\sigma' \prec_T \beta$ for all $1 \leq i \leq m$. By definition, $conv(\Gamma)$ is a confluent and terminating rewrite system. Since Γ is consistent, $t_j{\downarrow_{conv(\Gamma)}} \approx t_j$ is *β-true* in Γ for $1 \leq j \leq n$. Thus there exists a chain such that $L_i\sigma \to_{conv(\Gamma)} \cdots \to_{conv(\Gamma)} L_i\sigma'$ and $L_i\sigma'$ is *β-false* (*β-true* or *β-undefined*) in Γ. Now there also exists a chain $L_i\sigma \to_{conv(\Gamma)} \cdots \to_{conv(\Gamma)} L_i\sigma{\downarrow_{conv(\Gamma)}}$. By definition of convergence there must exist a chain $L_i\sigma' \to_{conv(\Gamma)} \cdots \to_{conv(\Gamma)} L_i\sigma{\downarrow_{conv(\Gamma)}}$. Thus $L_i\sigma{\downarrow_{conv(\Gamma)}} = L_i\sigma'{\downarrow_{conv(\Gamma)}}$. □

Auxiliary Lemma for the Proof of Lemma 9 and Theorem 2

Lemma 19 *Suppose a sound state $(\Gamma; N; U; \beta; k; \top)$ resulting from a regular run. If there exists a $C \in N \cup U$ and a grounding substitution σ such that $C\sigma$ is β-false in Γ, then Conflict is applicable. Otherwise, if there exists a $C \in N \cup U$ and a grounding substitution σ such that $C\sigma \prec_T \beta$ and there exists at least one $L \in C$ such that $L\sigma$ is β-undefined, then one of the rules Propagate or Decide is applicable and a β-undefined literal $K \in D$, where $D \in gnd_{\prec_T}\beta(N \cup U)$ is β-defined after application.*

Proof Let $(\Gamma; N; U; \beta; k; \top)$ be a state resulting from a regular run. Suppose there exists a $C \in N \cup U$ and a grounding σ such that $C\sigma$ is *β-false* in Γ, then by Lemma 18 there exists an irreducible substitution σ' such that $C\sigma'$ is *β-false*. Thus *Conflict* is applicable. Now suppose there exists a $C \in N \cup U$ and a grounding substitution σ such that $C\sigma \prec_T \beta$ and there exists at least one $L \in C$ such that $L\sigma$ is *β-undefined*. By Lemma 18 there exists a irreducible substitution σ' such that $L\sigma'$ is *β-undefined*. Now assume that $C = C_0 \vee C_1 \vee L$ such that $C_1\sigma' = L\sigma' \vee ... \vee L\sigma'$ and $C_0\sigma'$ is *β-false* in Γ. Then *Propagate* is applicable. Let $C_1 = L_1, ..., L_n$ and $\mu = mgu(L_1, ..., L_n, L)$. Now let $[I_1, ..., I_m]$ be the reduction chain application from Γ to $L\sigma'^{k:(L \vee C_0)\mu \cdot \sigma'}$. Let $I_m = (s_m \# t_m \cdot \sigma_m, (s_m \# t_m \vee C_m) \cdot \sigma_m, I_j, I_k, p_m)$. Then $L\sigma'{\downarrow_{conv(\Gamma)}} = s_m \# t_m \sigma_m$ by definition of a reduction chain application. Thus $L\sigma'$ is *β-true* in Γ, $s_m \# t_m \sigma_m$. Since $L\sigma{\downarrow_{conv(\Gamma)}} = L\sigma'{\downarrow_{conv(\Gamma)}}$ by Lemma 18, $L\sigma$ is *β-true* in Γ, $s_m \# t_m \sigma_m$ as well. If $C_0\sigma$ is *β-undefined* or *β-true* in Γ then Propagate is not applicable to $C\sigma'$. If Decide is not applicable by definition of a regular run, then there exists a clause $C' \in (N \cup U)$ and a substitution δ such that Propagate is applicable. Then we can apply Propagate by definition of a regular run and a previously undefined literal gets defined after application as seen above and we are done. Now suppose that there exists no such clause. Then let $[I'_1, ..., I'_l]$ be the reduction chain application from Γ to $L\sigma'^{k+1:C \cdot \sigma'}$ and $I'_l = (s_l \# t_l \cdot \sigma_l, (s_l \# t_l \vee C_l) \cdot \sigma_l, I'_j, I'_k, p_l)$. Then $L\sigma'{\downarrow_{conv(\Gamma)}} = (s_l \# t_l)\sigma_l$ by definition of a reduction chain application. Thus $L\sigma'$ is *β-true* in Γ, $(s_l \# t_l)\sigma_l$. Since $L\sigma{\downarrow_{conv(\Gamma)}} = L\sigma'{\downarrow_{conv(\Gamma)}}$ by lemma 18, $L\sigma$ is *β-true* in Γ, $(s_l \# t_l)\sigma_l$ as well. $(s_l \# t_l)\sigma_l^{k+1:(s_l \# t_l \vee comp(s_l \# t_l)) \cdot \sigma_l}$ can be added to Γ by definition of a regular run and also by definition of Decide since $C \in N \cup U$, σ' is grounding for C and irreducible in $conv(\Gamma)$, $L\sigma'$ is *β-undefined* in Γ and $C\sigma' \prec_T \beta$. □

Auxiliary Lemma for the Proof of Lemma 9

Lemma 20 *Suppose a sound state* $(\Gamma; N; U; \beta; k; D \cdot \sigma)$ *resulting from a regular run. Then* $D\sigma$ *is of level* 1 *or higher.*

Proof Let $(\Gamma; N; U; \beta; k; D \cdot \sigma)$ be a state resulting from a regular run. Suppose that $D\sigma$ is not of level 1 or higher, thus $D\sigma$ is of level 0. Then *Conflict* was applied earlier to a clause that was of level 1 or higher. Thus there must have been an application of *Explore-Refutation* on a state $(\Gamma, \Gamma', L; N; U; \beta; l; D' \cdot \sigma')$ between the state after the application of *Conflict* and the current state resulting in a state $(\Gamma, \Gamma', L^{l:(L \vee C)\cdot\delta}; N; U; \beta; l; D'' \cdot \sigma'')$ such that $D'\sigma'$ is of level l and $D''\sigma''$ is of level 0, since no other rule can reduce the level of $D'\sigma'$. Then there exists a $K \in D'\sigma'$ such that L is the defining literal of K. Let $[I_1, ..., I_m]$ be the refutation of K and $I_j = (s_j \# t_j \cdot \sigma_j, (s_j \# t_j \vee C_j) \cdot \sigma_j, I_i, I_k, p_j)$ be the step that was chosen by *Explore-Refutation*. Then $D''\sigma'' = (s_j \# t_j \vee C_j)\sigma_j$. $C\delta \subset C_j\sigma_j$ has to hold since L is the defining literal of K. Then $C\delta$ must be of level 0 or empty. Note that $C\delta$ is of level l if L is a decision literal. But then, by the definition of a regular run, $L^{l:(L \vee C)\cdot\delta}$ must have been propagated before the first decision, since propagation is exhaustive at level 0. Contradiction. □

Proof of Lemma 9 If a regular run (without rule Grow) ends in a stuck state $(\Gamma; N; U; \beta; k; D)$, then $D = \top$ and all ground literals $L\sigma \prec_T \beta$, where $L \vee C \in N \cup U$ are β-*defined* in Γ.

Proof First we prove that stuck states never appear during conflict resolution. Assume a sound state $(\Gamma; N; U; \beta; k; D \cdot \sigma)$ resulting from a regular run. Now we show that we can always apply a rule. Suppose that $D\sigma = (D' \vee L \vee L')\sigma$ such that $L\sigma = L'\sigma$. Then we must apply *Factorize* by the definition of a regular run. Now suppose that *Factorize* is not applicable and $\Gamma{:=}\Gamma', L$ and $D\sigma$ is false in Γ'. If $D\sigma = (D' \vee s \not\approx s')\sigma$ such that $s\sigma = s'\sigma$, we can apply *Equality-Resolution*. So suppose that *Equality-Resolution* is not applicable. Then we can apply *Skip*. Now suppose that $\Gamma{:=}\Gamma', L^{k:(L \vee C)\delta}$ and L is the defining literal of at least one literal in $D\sigma$, so Skip is not applicable. If $D\sigma = (D' \vee L')\sigma$ where $D'\sigma$ is of level $i < k$ and $L'\sigma$ is of level k and *Skip* was applied at least once during this conflict resolution, then Backtrack is applicable. If *Skip* was not applied and $L = comp(L'\sigma)$ and L is a decision literal, then Backtrack is also applicable. Otherwise, let $(s \# t)\sigma \in D\sigma$ such that $K \prec_{\Gamma^*} (s \# t)\sigma$ for all $K \in D\sigma$. $(s \# t)\sigma$ exists since Factorize is not applicable. By Lemma 20, $(s \# t)\sigma$ must be of level 1 or higher. By the definition of $\prec_{\Gamma^*}$, L must be the defining literal of $(s \# t)\sigma$ since L is of level 1 or higher and any literal in $D\sigma$ that has another defining literal is smaller than $(s \# t)\sigma$. Now suppose that L is a decision literal and $(s \# t)\sigma = comp(L)$. Then $(s \# t)\sigma$ is of level k and all other literals $K \in D\sigma$ are of level $i < k$, since $(s \# t)\sigma$ is the smallest β-*false* literal of level k and Factorize is not applicable. In this case Explore-Refutation is not applicable since a paramodulation step with the decision literal does not make the conflict clause smaller. But Backtrack is applicable in this case even if Skip was not applied earlier by the definition of a regular run. Thus $(s \# t)\sigma \neq comp(L)$ or L is a propagated literal has to hold. We show that in this case Explore-Refutation is applicable. Let $[I_1, ..., I_m]$ be a refutation of $(s \# t)\sigma$ from Γ, $I_m = (s_m \# t_m \cdot \sigma_m, (s_m \# t_m \vee C_m) \cdot \sigma_m, I_j, I_k, p_m)$. Since $[I_1, ..., I_m]$ is a refutation $s_m \# t_m\sigma_m = s' \not\approx s'$. Furthermore any I_i either contains a clause annotation from Γ, $(s \# t)\sigma^{k:D\cdot\sigma}$ or it is a rewrite inference from $I_{j'}, I_{k'}$ with $j', k' < i$. Thus by Lemma 3 it inductively follows that $C_m\sigma_m = D'\sigma_m \vee ... \vee D'\sigma_m \vee C'_1\sigma_m \vee ... \vee C'_n\sigma_m$, where $C'_1\sigma_m, ..., C'_n\sigma_m$ are clauses from Γ without the leading trail literal and $D\sigma = D'\sigma_m \vee (s \# t)\sigma$. Since L is the defining literal of $(s \# t)\sigma$ there must exist at least one C'_i such that $C'_i\sigma_m = C\delta$. If L is a propagated literal, then

any literal in $C'_i\sigma_m$ is smaller than $(s \# t)\sigma$, since they are already *false* in Γ'. If L is a decision literal, then $C'_i\sigma_m = comp(L)$. Then $comp(L)$ is smaller, since $(s \# t)\sigma \neq comp(L)$ and $(s \# t)\sigma \neq L$. Thus $comp(L) \prec_{\Gamma^*} (s \# t)\sigma$. Any other literal in $C_1\sigma_m, ..., C'_n\sigma_m$ is smaller in $\prec_{\Gamma^*}$, since they are already defined in Γ'. Since $Factorize$ is not applicable $(s \# t)\sigma$ is also strictly maximal in $D'\sigma_m$. Thus $(s_m \# t_m \vee C_m)\sigma_m \prec_{\Gamma^*} D\sigma$ which makes *Explore-Refutation* applicable.
Now by Lemma 19 it holds that if there exists an *β-undefined* literal in $gnd_{\prec_T \beta}(N \cup U)$, we can always apply at least one of the rules *Propagate* or $Decide$ which makes a previously *β-undefined* literal in $gnd_{\prec_T \beta}(N \cup U)$ *β-defined*. □

Proof of Lemma 10 Suppose a sound state $(\Gamma; N; U; \beta; k; D)$ resulting from a regular run where $D \notin \{\top, \bot\}$. If $Backtrack$ is not applicable then any set of applications of *Explore-Refutation*, $Skip$, *Factorize*, *Equality-Resolution* will finally result in a sound state $(\Gamma'; N; U; \beta; k; D')$, where $D' \prec_{\Gamma^*} D$. Then Backtrack will be finally applicable.

Proof Assume a sound state $(\Gamma; N; U; \beta; k; D \cdot \sigma)$ resulting from a regular run. Let $(s \# t)\sigma \in D\sigma$ such that $L \preceq_{\Gamma^*} (s \# t)\sigma$ for all $L \in D\sigma$. If $(s \# t)\sigma$ occurs twice in $D\sigma$, then $Factorize$ is applicable. Suppose that it is applied. Then $D\sigma = (D' \vee (s \# t) \vee L)\sigma$, where $L\sigma = (s \# t)\sigma$. Then $\mu = mgu(s \# t, L)$ and the new conflict clause is $(D' \vee s \# t)\mu\sigma \prec_{\Gamma^*} D\sigma$. Thus in this case we are done. If $Factorize$ is not applicable, then the only remaining applicable rules are $Skip$, *Explore-Refutation* and *Equality-Resolution*. If $\Gamma = \Gamma', L, \Gamma''$ where L is the defining literal of $(s \# t)\sigma$, then $Skip$ is applicable $|\Gamma''|$ times, since otherwise $(s \# t)\sigma$ would not be maximal in $D\sigma$. So at some point it is no longer applicable. Since $D\sigma$ is finite, *Equality-Resolution* can be applied only finitely often. Thus we finally have to apply *Explore-Refutation*. Then $[I_1, ..., I_m]$ is a refutation of $(s \# t)\sigma$ from Γ, and there exists an $1 \leq j \leq m$, such that $I_j = (s_j \# t_j \cdot \sigma_j, (s_j \# t_j \vee C_j) \cdot \sigma_j, I_l, I_k, p_j), (C_j \vee s_j \# t_j)\sigma_j \prec_{\Gamma^*} (D' \vee s \# t)\sigma$. Otherwise *Explore-Refutation* would not be applicable, contradicting Lemma 9. Thus in this case we are done.

Now we show that Backtrack is finally applicable. Since $\prec_{\Gamma^*}$ is well-founded and Γ is finite there must be a state where *Explore-Refutation*, $Skip$, *Factorize*, *Equality-Resolution* are no longer applicable. By Lemma 20 the conflict clause in this state must be of level 1 or higher, thus $\bot$ cannot be inferred. Suppose that it is always of level $i \geq l$ for some l. The smallest literal of level l that is *false* in Γ is $comp(L)$, where L is the decision literal of level l. Since we can always reduce if *Backtrack* is not applicable and since we can always apply a rule by Lemma 9, we must finally reach a conflict clause $comp(L) \vee C$, where C is of level $j < l$. Thus *Backtrack* is applicable. □

Proof of Lemma 11 Let N be a set of clauses and β be a ground term. Then any regular run that never uses Grow terminates.

Proof Assume a new ground clause $D\sigma$ is learned. By Lemma 6 all learned clauses are non-redundant. Thus $D\sigma$ is non-redundant. By the definition of a regular run *Factorize* has precedence over all other rules. Thus $D\sigma$ does not contain any duplicate literals. By Theorem 1, $D\sigma \prec_T \beta$ has to hold. There are only finitely many clauses $C\sigma \prec_T \beta$, where $C\sigma$ is neither a tautology nor does it contain any duplicate literals. Thus there are only finitely many clauses $D\sigma$ that can be learned. Thus there are only finitely many literals that can be decided or propagated. □

Proof of Lemma 12 If a regular run reaches the state $(\Gamma; N; U; \beta; k; \bot)$ then N is unsatisfiable.

Proof By definition of soundness, all learned clauses are consequences of $N \cup U$, Definition 13.5, and Γ is satisfiable, Definition 13.1. □

Proof of Theorem 2 Let N be an unsatisfiable clause set, and $\prec_T$ a desired term ordering. For any ground term β where $gnd_{\prec_T \beta}(N)$ is unsatisfiable, any regular SCL(EQ) run without rule Grow will terminate by deriving $\bot$.

Proof Since regular runs of SCL(EQ) terminate we just need to prove that it terminates in a failure state. Assume by contradiction that we terminate in a state $(\Gamma; N; U; \beta; k; \top)$. If no rule can be applied in Γ then for all $s \# t \in C$ for some arbitrary $C \in gnd_{\prec_T \beta}(N)$ it holds that $s \# t$ is *β-defined* in Γ (otherwise Propagate or Decide woud be applicable, see Lemma 19) and there aren't any clauses in $gnd_{\prec_T \beta}(N)$ *β-false* under Γ (otherwise *Conflict* would be applicable, see again Lemma 19). Thus, for each $C \in gnd_{\prec_T \beta}(N)$ it holds that C is *β-true* in Γ. So we have $\Gamma \models gnd_{\prec_T \beta}(N)$, but by hypothesis there is a superposition refutation of N that only uses ground literals from $gnd_{\prec_T \beta}(N)$, so also $gnd_{\prec_T \beta}(N)$ is unsatisfiable, a contradiction. □

Lemma 21 *(Only Non-Redundant Clauses Building the Trail) Let* $\Gamma = [L_1^{i_1:C_1 \cdot \sigma_1}, ..., L_n^{i_n:C_n \cdot \sigma_n}]$ *be a trail. If* $L_j^{i_j:C_j \cdot \sigma_j}$ *is a propagated literal and there exist clauses* $\{D_1 \vee K_1, ..., D_m \vee K_m\}$ *with grounding substitutions* $\delta_1, ..., \delta_m$ *such that* $N := \{(D_1 \vee K_1)\delta_1, ..., (D_m \vee K_m)\delta_m\} \prec_{\Gamma^*} C_j\sigma_j$ *and* $\{(D_1 \vee K_1)\delta_1, ..., (D_m \vee K_m)\delta_m\} \models C_j\sigma_j$*, then there exists a* $(D_k \vee K_k)\delta_k \in N$ *such that*

$$[L_1^{i_1:C_1 \cdot \sigma_1}, ..., L_{j-1}^{i_{j-1}:C_{j-1} \cdot \sigma_{j-1}}, K_k^{i_j:(D_k \vee K_k) \cdot \delta_k}, ..., L_n^{i_n:C_n \cdot \sigma_n}]$$

is a trail.

Proof Let $N = \{(D_1 \vee K_1)\delta_1, ..., (D_m \vee K_m)\delta_m\}$ and $L_j^{i_j:C_j \cdot \sigma_j}$ be as above. Let $\Gamma' = [L_1^{i_1:C_1 \cdot \sigma_1}, ..., L_{j-1}^{i_{j-1}:C_{j-1} \cdot \sigma_{j-1}}]$. Now suppose that for every literal $L \in N$ it holds $L \prec_{\Gamma^*} L_j$. Then every literal in N is defined in Γ' and $\Gamma' \models N$, otherwise *Conflict* would have been applied to a clause in N. Thus $\Gamma' \models C_j\sigma_j$ would have to hold as well. But by definition of a trail L_j is undefined in Γ'. Thus there must be at least one clause $(D_k \vee K_k)\delta_k \in N$ with $K_k = L_j$ and $D_k\delta_k \prec_{\Gamma^*} L_j$ (otherwise $(D_k \vee K_k)\delta_k \not\prec_{\Gamma^*} C_j\sigma_j$), such that $\Gamma' \not\models D_k$. Suppose that $\Gamma' \models D_k$. Then $N \not\models C_j\sigma_j$, since there exists an allocation, namely Γ', $\neg L_k$ such that $\Gamma', \neg L_k \models N$ but $\Gamma', \neg L_k \not\models C_j\sigma_j$. Thus we can replace $L_j^{i_j:C_j \cdot \sigma_j}$ by $K_k^{i_j:(D_k \vee K_k) \cdot \delta_k}$ in Γ. □

Further Examples

For the following examples we assume a term ordering $\prec_{kbo}$, unique weight 1 and with precedence $d \prec c \prec b \prec a \prec a_1 \prec \cdots \prec a_n \prec g \prec h \prec f$. Further assume β to be large enough.

Example 6 (Implicit Conflict after Decision) Consider the following clause set N

$$C_1 := h(x) \approx g(x) \vee c \approx d \qquad C_2 := f(x) \approx g(x) \vee a \approx b$$
$$C_3 := f(x) \not\approx h(x) \vee f(x) \not\approx g(x)$$

Suppose we apply the rule Decide first to C_1 and then to C_2 with substitution $\sigma = \{x \to a\}$. Then we yield a conflict with $C_3\sigma$, resulting in the following state:

$$([h(a) \approx g(a)^{1:(h(x) \approx g(x) \vee h(x) \not\approx g(x)) \cdot \sigma}, f(a) \approx g(a)^{2:(f(x) \approx g(x) \vee f(x) \not\approx g(x)) \cdot \sigma}];$$
$$N; \{\}; 2; C_3 \cdot \sigma)$$

According to $\prec_{\Gamma^*}$, $f(a) \not\approx h(a)$ is the greatest literal in $C_3\sigma$. Since $f(a) \approx g(a)$ is the defining literal of $f(a) \not\approx h(a)$ we can not apply Skip. Factorize is also not applicable, since $f(a) \not\approx h(a)$ and $f(a) \not\approx g(a)$ are not equal. Thus we must apply Explore-Refutation to the greatest literal $f(a) \not\approx h(a)$. The rule first creates a refutation $[I_1, ..., I_5]$, where:

$$
\begin{aligned}
I_1 &:= ((f(x) \not\approx h(x)) \cdot \sigma, C_3 \cdot \sigma, \epsilon, \epsilon, \epsilon)\\
I_2 &:= ((f(x) \approx g(x)) \cdot \sigma, (f(x) \approx g(x) \vee f(x) \not\approx g(x)) \cdot \sigma, \epsilon, \epsilon, \epsilon)\\
I_3 &:= ((h(x) \approx g(x)) \cdot \sigma, (h(x) \approx g(x) \vee h(x) \not\approx g(x)) \cdot \sigma, \epsilon, \epsilon, \epsilon)\\
I_4 &:= ((h(x) \not\approx g(x)) \cdot \sigma, (h(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee f(x) \not\approx g(x)) \cdot \sigma, I_2, I_1, 1)\\
I_5 &:= ((g(x) \not\approx g(x)) \cdot \sigma, (g(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee f(x) \not\approx g(x)\\
&\quad \vee h(x) \not\approx g(x)) \cdot \sigma, I_4, I_3, 1)
\end{aligned}
$$

Explore-Refutation can now choose either I_4 or I_5. Both, $(h(x) \not\approx g(x))\sigma$ and $(g(x) \not\approx g(x))\sigma$ are smaller than $(f(x) \not\approx h(x))\sigma$ according to $\prec_{\Gamma^*}$ and false in Γ. Suppose we choose I_5. Now our new conflict state is:

$$
\begin{aligned}
&([h(a) \approx g(a)^{1:(h(x)\approx g(x) \vee h(x) \not\approx g(x)) \cdot \sigma}, f(a) \approx g(a)^{2:(f(x)\approx g(x) \vee f(x) \not\approx g(x)) \cdot \sigma}];\\
&N; \{\}; 2; (g(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee f(x) \not\approx g(x) \vee h(x) \not\approx g(x))\sigma).
\end{aligned}
$$

Now we apply *Equality-Resolution* and Factorize to get the new state

$$
\begin{aligned}
&([g(a) \approx h(a)^{1:(g(x)\approx h(x) \vee g(x) \not\approx h(x)) \cdot \sigma}, f(a) \approx g(a)^{2:(f(x)\approx g(x) \vee f(x) \not\approx g(x)) \cdot \sigma}];\\
&N; \{\}; 2; (f(x) \not\approx g(x) \vee h(x) \not\approx g(x)) \cdot \sigma)
\end{aligned}
$$

Now we can backtrack. Note, that this clause is non-redundant according to our ordering, although conflict was applied immediately after decision.

Example 7 (Undefined conflict clause) Consider the following ground clause set N:

$$C_1 := f(a, a) \not\approx f(b, b) \vee c \approx d \qquad C_2 := a \approx b \vee f(a, a) \approx f(b, b)$$

Suppose that we decide $f(a, a) \not\approx f(b, b)$. Then C_2 is false in Γ. Conflict state is as follows: $([f(a, a) \not\approx f(b, b)^{1:f(a,a) \not\approx f(b,b) \vee f(a,a) \approx f(b,b)}]; N; \{\}; 2; C_2)$. Explore-Refutation creates the following ground refutation for $a \approx b$, since it is greatest literal in the conflict clause:

$$
\begin{aligned}
I_1 &:= (f(a, a) \not\approx f(b, b), f(a, a) \not\approx f(b, b) \vee f(a, a) \approx f(b, b), \epsilon, \epsilon, \epsilon)\\
I_2 &:= (a \approx b, C_2, \epsilon, \epsilon, \epsilon)\\
I_3 &:= (f(b, a) \not\approx f(b, b), f(b, a) \not\approx f(b, b) \vee f(a, a) \approx f(b, b) \vee f(a, a) \approx f(b, b),\\
&\quad I_2, I_1, 11)\\
I_4 &:= (f(b, b) \not\approx f(b, b), f(b, b) \not\approx f(b, b) \vee f(a, a) \approx f(b, b) \vee f(a, a) \approx f(b, b)\\
&\quad \vee f(a, a) \approx f(b, b), I_3, I_1, 12)
\end{aligned}
$$

As one can see, the intermediate result $f(b, a) \not\approx f(b, b)$ is not false in Γ. Thus it is no candidate for the new conflict clause. We have to choose I_4. The new state is thus:

$$
\begin{aligned}
&([f(a, a) \not\approx f(b, b)^{1:f(a,a) \not\approx f(b,b) \vee f(a,a) \approx f(b,b)}]; N; \{\}; 2;\\
&f(b, b) \not\approx f(b, b) \vee f(a, a) \approx f(b, b) \vee f(a, a) \approx f(b, b) \vee f(a, a) \approx f(b, b))
\end{aligned}
$$

Now we can apply Equality-Resolution and two times Factorize to get the final clause $f(a, a) \approx f(b, b)$ with which we can backtrack.

Example 8 (SCL(EQ) vs. Superposition: Clause learning) Assume clauses:

$$\begin{aligned} C_1 &:= b \approx c \vee c \approx d \\ C_2 &:= a_1 \approx b \vee a_1 \approx c \\ &\ldots \\ C_{n+1} &:= a_n \approx b \vee a_n \approx c \end{aligned}$$

The completeness proof of superposition requires that adding a new literal to an interpretation does not make any smaller literal true. In this example, however, after adding $b \approx c$ to the interpretation, we cannot any further literal, since it breaks this invariant. So in superposition we would have to add the following clauses with the help of the $EqualityFactoring$ rule:

$$\begin{aligned} C_{n+2} &:= b \not\approx c \vee a_1 \approx c \\ &\ldots \\ C_{2n+1} &:= b \not\approx c \vee a_n \approx c \end{aligned}$$

In SCL(EQ) on the other hand we can just decide a literal in each clause to get a model for this clause set. As we support undefined literals we do not have to bother with this problem at all. For example if we add $b \approx c$ to our model, both literals $a_1 \approx b$ and $a_1 \approx c$ are undefined in our model. Thus we need to decide one of these literals to add it to our model.

References

1. Alagi, G., Weidenbach, C.: NRCL - A model building approach to the Bernays-Schönfinkel fragment. In: Lutz, C., Ranise, S. (eds.) Frontiers of Combining Systems—10th International Symposium, FroCoS 2015, Wroclaw, Poland, September 21–24, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9322, pp. 69–84. Springer, Cham. https://doi.org/10.1007/978-3-319-24246-0_5 (2015)
2. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994). https://doi.org/10.1093/logcom/4.3.217
3. Bachmair, L., Ganzinger, H., Waldmann, U.: Superposition with simplification as a decision procedure for the monadic class with equality. In: Gottlob, G., Leitsch, A., Mundici, D. (eds.) Computational Logic and Proof Theory, Third Kurt Gödel Colloquium. LNCS, vol. 713, pp. 83–96. Springer, Berlin. https://doi.org/10.1007/BFb0022557 (1993)
4. Bachmair, L., Ganzinger, H., Voronkov, A.: Elimination of equality via transformation with ordering constraints. In: Kirchner, C., Kirchner, H. (eds.) International Conference on Automated Deduction. Lecture Notes in Computer Science, vol. 1421, pp. 175–190. Springer, Berlin. https://doi.org/10.1007/BFb0054259 (1998)
5. Baumgartner, P.: Hyper tableau—the next generation. In: de Swart, H.C.M. (ed.) Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX '98, Oisterwijk, The Netherlands, May 5–8, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1397, pp. 60–76. Springer, Berlin. https://doi.org/10.1007/3-540-69778-0_14 (1998)
6. Baumgartner, P., Tinelli, C.: The model evolution calculus with equality. In: Nieuwenhuis, R. (ed.) 20th International Conference on Automated Deduction. LNAI, vol. 3632, pp. 392–408. Springer, Berlin. https://doi.org/10.1007/11532231_29 (2005)
7. Baumgartner, P., Waldmann, U.: Superposition and model evolution combined. In: Schmidt, R.A. (ed.) Automated Deduction—CADE-22. LNAI, vol. 5663, pp. 17–34. Springer, Berlin. https://doi.org/10.1007/978-3-642-02959-2_2 (2009)
8. Baumgartner, P., Fuchs, A., Tinelli, C.: Lemma learning in the model evolution calculus. In: Hermann, M., Voronkov, A. (eds.) 13th International Conference, LPAR 2006. LNAI, vol. 4246, pp. 572–586. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11916277_39 (2006)
9. Baumgartner, P., Furbach, U., Pelzer, B.: Hyper tableaux with equality. In: Pfenning, F. (ed.) International Conference on Automated Deduction. LNAI, vol. 4603, pp. 492–507. Springer, Berlin. https://doi.org/10.1007/978-3-540-73595-3_36 (2007)
10. Baumgartner, P., Pelzer, B., Tinelli, C.: Model evolution with equality-revised and implemented. J. Symb. Comput. **47**(9), 1011–1045 (2012). https://doi.org/10.1016/j.jsc.2011.12.031

11. Bayardo, R.J., Schrag, R.: Using CSP look-back techniques to solve exceptionally hard SAT instances. In: Freuder, E.C. (ed.) Proceedings of the Second International Conference on Principles and Practice of Constraint Programming, Cambridge, Massachusetts, USA, August 19–22, 1996. LNCS, vol. 1118, pp. 46–60. Springer, Berlin. https://doi.org/10.1007/3-540-61551-2_65 (1996)
12. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Amsterdam (2009)
13. Bonacina, M.P., Plaisted, D.A.: SGGS theorem proving: an exposition. In: Schulz, S., Moura, L.D., Konev, B. (eds.) PAAR-2014. 4th Workshop on Practical Aspects of Automated Reasoning. EPiC Series in Computing, vol. 31, pp. 25–38. EasyChair, Bramhall. https://doi.org/10.29007/m2vf (2015)
14. Bonacina, M.P., Furbach, U., Sofronie-Stokkermans, V.: In: Martí-Oliet, N., Ölveczky, P.C., Talcott, C. (eds.) On First-Order Model-Based Reasoning. LNAI, vol. 9200, pp. 181–204. Springer, Cham. https://doi.org/10.1007/978-3-319-23165-5_8 (2015)
15. Bromberger, M., Fiori, A., Weidenbach, C.: Deciding the Bernays–Schoenfinkel fragment over bounded difference constraints by simple clause learning over theories. In: Henglein, F., Shoham, S., Vizel, Y. (eds.) Verification, Model Checking, and Abstract Interpretation—22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17–19, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12597, pp. 511–533. Springer, Cham. https://doi.org/10.1007/978-3-030-67067-2_23 (2021)
16. Bromberger, M., Gehl, T., Leutgeb, L., Weidenbach, C.: A two-watched literal scheme for first-order logic. In: Boris Konev, A.S. Claudia Schon (ed.) Proceedings of the Workshop on Practical Aspects of Automated Reasoning Co-located with the 11th International Joint Conference on Automated Reasoning (FLoC/IJCAR 2022). CEUR Workshop Proceedings, vol. 3201. CEUR-WS.org, RWTH Aachen, Ahornstr. 55, 52056 Aachen (2022)
17. Bromberger, M., Schwarz, S., Weidenbach, C.: SCL(FOL) Revisited. Preprint at http://arxiv.org/2302.05954 (2023)
18. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3), 201–215 (1960). https://doi.org/10.1145/321033.321034
19. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM **5**(7), 394–397 (1962). https://doi.org/10.1145/368273.368557
20. Dershowitz, N., Plaisted, D.A.: Rewriting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 535–610. Elsevier, Berlin (2001)
21. Fiori, A., Weidenbach, C.: SCL clause learning from simple models. In: Fontaine, P. (ed.) 27th International Conference on Automated Deduction, CADE-27. LNAI, vol. 11716. Springer, Cham. https://doi.org/10.1007/978-3-030-29436-6_14 (2019)
22. Gallier, J., Narendran, P., Plaisted, D., Raatz, S., Snyder, W.: An algorithm for finding canonical sets of ground rewrite rules in polynomial time. J. ACM **40**(1), 1–16 (1993). https://doi.org/10.1145/138027.138032
23. Ganzinger, H., de Nivelle, H.: A superposition decision procedure for the guarded fragment with equality. In: LICS, pp. 295–304. https://doi.org/10.1109/LICS.1999.782624 (1999)
24. Gleiss, B., Kovács, L., Rath, J.: Subsumption demodulation in first-order theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning—10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12166, pp. 297–315. Springer, Cham. https://doi.org/10.1007/978-3-030-51074-9_17 (2020)
25. Korovin, K.: In: Voronkov, A., Weidenbach, C. (eds.) Inst-Gen—A Modular Approach to Instantiation-Based Automated Reasoning, pp. 239–270. Springer, Berlin. https://doi.org/10.1007/978-3-642-37651-1_10 (2013)
26. Korovin, K., Sticksel, C.: iProver-Eq: An instantiation-based theorem prover with equality. In: Giesl, J., Hähnle, R. (eds.) 5th International Joint Conference, IJCAR 2010. LNAI, vol. 6173, pp. 196–202. Springer, Berlin. https://doi.org/10.1007/978-3-642-14203-1_17 (2010)
27. Leidinger, H., and, C.W.: SCL(EQ): SCL for first-order logic with equality. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning—11th International Joint Conference, IJCAR 2022 Held as Part of the Federated Logic Conference, Haifa, Israel, August 8–10, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13385, pp. 228–247. Springer, Cham. https://doi.org/10.1007/978-3-031-10769-6_14 (2022)
28. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: engineering an efficient SAT solver. In: Design Automation Conference, 2001. Proceedings, pp. 530–535. ACM, New York. https://doi.org/10.1145/378239.379017 (2001)
29. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. J. ACM **27**(2), 356–364 (1980). https://doi.org/10.1145/322186.322198
30. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving sat and sat modulo theories: from an abstract Davis–Putnam–Logemann–Loveland procedure to dpll(t). J. ACM **53**, 937–977 (2006)

31. Plaisted, D.A., Zhu, Y.: Ordered semantic hyper-linking. J. Autom. Reason. **25**(3), 167–217 (2000). https://doi.org/10.1023/A:1006376231563
32. Robinson, G., Wos, L.: Paramodulation and theorem-proving in first-order theories with equality. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence 4, pp. 135–150. https://doi.org/10.1007/978-3-642-81955-1_19 (1969)
33. Silva, J.P.M., Sakallah, K.A.: GRASP—a new search algorithm for satisfiability. In: International Conference on Computer Aided Design, ICCAD, pp. 220–227. IEEE Computer Society Press, Boston. https://doi.org/10.1007/978-1-4615-0292-0_7 (1996)
34. Sutcliffe, G.: The TPTP problem library and associated infrastructure—from CNF to th0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017). https://doi.org/10.1007/s10817-017-9407-7
35. Teucke, A.: An approximation and refinement approach to first-order automated reasoning. Doctoral thesis, Saarland University. https://doi.org/10.22028/D291-27196 (2018)
36. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning—10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12166, pp. 316–334. Springer, Cham. https://doi.org/10.1007/s10817-022-09621-7 (2020)
37. Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 1965–2012. Elsevier, Hoboken (2001)
38. Wischnewski, P.: Efficient reasoning procedures for complex first-order theories. PhD thesis, Saarland University. https://doi.org/10.22028/D291-26406 (2012)
39. Weidenbach, C.: Automated reasoning building blocks. In: Meyer, R., Platzer, A., Wehrheim, H. (eds.) Correct System Design—Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Oldenburg, Germany, September 8–9, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9360, pp. 172–188. Springer, Cham (2015)
40. Weidenbach, C., Wischnewski, P.: Contextual rewriting in SPASS. In: PAAR/ESHOL. CEUR Workshop Proceedings, vol. 373, pp. 115–124 (2008)
41. Weidenbach, C., Wischnewski, P.: Subterm contextual rewriting. AI Commun. **23**(2–3), 97–109 (2010). https://doi.org/10.3233/AIC-2010-0459