Exploring Partial Models with SCL

Martin Bromberger, Simon Schwarz, Christoph Weidenbach

EPiC Series in Computing

Volume 94, 2023, Pages 48–72

Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Exploring Partial Models with SCL

Martin Bromberger[1], Simon Schwarz[1,2], and Christoph Weidenbach[1]

[1] Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
[2] Graduate School of Computer Science, Saarbrücken, Germany

Abstract

The family of SCL (Clause Learning from Simple Models) calculi learns clauses with respect to a partial model assumption, similar to CDCL (Conflict Driven Clause Learning). The partial model always consists of ground first-order literals and is built by decisions and propagations. In contrast to propositional logic where propagation chains are always finite, in first-order logic they can become infinite. Therefore, the SCL family does not require exhaustive propagation and the size of the partial model is always finitely bounded. Any partial model not leading to a conflict constitutes a model for the respective finitely bounded ground clause set. We show that all potential partial models can be explored as part of the SCL calculus for first-order logic without equality and that any overall model is an extension of a partial model considered. Furthermore, SCL turns into a semi-decision procedure for first-order logic by extending the finite bound for any partial model not leading to a conflict.

1 Introduction

By now, there are three instances of the SCL calculus family: SCL for first-order logic without equality [13, 10], SCL for first-order logic over theories [8], and SCL for first-order logic with equality [16]. They share: (i) an explicit trail (partial model assumption) built from ground literals, (ii) a finite limit to the potential size of trails and hence considered ground instances, and (iii) non-redundant clause learning. The finite limit to the trail's size is a way to deal with potentially infinite propagations in first-order logic. For example from a trail $[P(a)]$ and a single clause $\neg P(x) \vee P(g(x))$ already infinitely many ground literals $P(g^i(a))$ can be propagated. Even in first-order fragments having a finite model property, the number of propagations may grow exponentially in the size of the input [20, 13]. Posing a finite limit on trail size let the SCL calculi run into *stuck* states. In a stuck state, the partial model assumption on the trail forms a model for the finitely many ground instances of a clause set that are smaller than the imposed limit. However, the partial model assumption is not necessarily a model for the clause set in general. In this paper, we present two approaches for dealing with such stuck states, namely exhaustive partial model exploration and fair increasing of model bounds.

Exhaustive partial model exploration First, we show that the search for a refutation as considered in previous work [13, 8, 16, 10] can be combined with an *exhaustive search* for all partial ground models under the current finite limit. We finally prove that in fact for any

model of the overall clause set, if it exists, our exhaustive search will yield the restriction of this model to the current finite limit, Theorem 19. Therefore, the SCL family enables simultaneous search for a refutation and a model in a controlled way. The general idea of the new HSCL calculus is to learn a new *excluding clause* from any stuck state that prevents the repetition of the stuck state. The family of propositional CDCL calculi uses similar ideas if extended to optimization [4]. For example, if a satisfying assignment with "minimal weight" should be computed, already found assignments not improving the weight are ruled out by learning respective clauses, e.g., a clause consisting of the negation of all decisions leading to the assignment. In HSCL, the learned excluding clauses are strictly distinguished from regular learned clauses. This strict separation is needed to recognize "regularly learned" clauses that are not dependent on any excluding clause. Such clauses are a direct consequence of the input clause set and can be kept indefinitely. In contrast, excluding clauses (and all clauses that are derived from excluding clauses) are not entailed by the original clause set and need to be invalidated at the end of exploration. Hence, our strict separation allows us to keep clauses that are not dependent on any excluding clause for future search. In contrast, if the excluding clauses are simply added as assumptions, all clauses learned under this assumption must be invalidated at the end of exploration.

In first-order logic there have been calculi developed that built an explicit model assumption, e.g. [2, 23, 3, 21, 5, 7], but to the best of our knowledge there is no calculus that learns new non-redundant clauses simultaneously towards a refutation and exhaustive model exploration. Our partial model candidates are by definition always finite domain models because the trail consists of ground literals. Finite model finding has a long-standing tradition in first-order logic reasoning [25, 24, 17, 18, 11, 22, 15] starting with the systems MACE [18, 17] and Finder [25] that explicitly search for a finite model and not for a refutation.

Increasing bounds As a second option in stuck states, we present a *fair* way of increasing the model bounds, enabling HSCL to search for larger partial model assumptions. By sufficiently extending bounds, this allows to exclude partial model candidates that are actually not extendable to an overall model, see Theorem 25. Moreover, HSCL can even explicitly learn clauses that exclude such non-extendable partial models from future searches, Theorem 27. By increasing model bounds in a fair way, refutational completeness of HSCL for first-order logic is achieved, see Theorem 26.

The paper is now organized as follows: After clarifying some notions, Section 2, the HSCL calculus is introduced in Section 3. The HSCL calculus is an extension of the already existing calculi for first-order logic without equality [13, 8, 10]. The paper ends with a discussion and directions to future work. This paper is an extension of a previously published workshop paper [9]. In this extended version, the technique of increasing bounds (Section 3.3) has been added. Note that all proofs are available in the appendix.

2 Preliminaries

The general notation follows the SCL(FOL) papers [13, 10].

We assume a first-order language without equality where N denotes a clause set; C, D denote clauses; L, K, H denote literals; A, B denote atoms; P, Q, R denote predicates; t, s terms; f, g, h function symbols; a, b, c constants; and x, y, z variables. Atoms, literals, clauses and clause sets are considered as usual, where in particular clauses are identified both with their disjunction and multiset of literals. The complement of a literal is denoted by the function comp. Semantic entailment $\models$ is defined as usual where variables in clauses are assumed to

be universally quantified. Substitutions σ, τ are total mappings from variables to terms, where $\mathrm{dom}(\sigma) := \{x \mid x\sigma \neq x\}$ is finite and $\mathrm{codom}(\sigma) := \{t \mid x\sigma = t, x \in \mathrm{dom}(\sigma)\}$. Their application is extended to literals, clauses, and sets of such objects in the usual way. A term, atom, clause, or a set of these objects is *ground* if it does not contain any variable. A substitution σ is *ground* if $\mathrm{codom}(\sigma)$ 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. The function mgu denotes the *most general unifier* of two terms, atoms, literals. We assume that any mgu of two terms or literals does not introduce any fresh variables and is idempotent. A *closure* is denoted as $C \cdot \sigma$ and is a pair of a clause C and a grounding substitution σ. The function gnd returns the set of all ground instances of a literal, clause, or clause set with respect to the signature of the respective clause set.

Let $\prec$ denote a well-founded, total, strict ordering on ground literals. This ordering is then 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 $C\sigma \prec D\sigma$. We define $\preceq$ as the reflexive closure of $\prec$ and $N^{\preceq C} := \{D \mid D \in N \text{ and } D \preceq C\}$.

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

For the sake of bounding, let $\prec_B$ denote a well-founded, total, strict ordering on ground atoms such that for any ground atom A there are only finitely many ground atoms B with $B \prec_B A$ [9]. For example, an instance of such an ordering could be KBO without zero-weight symbols and a strict, total precedence on symbols. The ordering $\prec_B$ is lifted to literals by comparing the respective atoms. It is lifted to clauses by a multiset extension. Given an ordering $\prec_B$ and a ground literal β, the function $\mathrm{gnd}^{\prec_B \beta}$ computes the set of all ground instances of a literal, clause, or clause set where the grounding is restricted to produce literals L with $L \prec_B \beta$. Given these restrictions, we can define partial models for literals $\prec_B \beta$.

Definition 2 (Partial models)**.** *A partial model M is a satisfiable set of ground literals. If $L \in M$, the ground literal L is interpreted as true. Conversely, if* $\mathrm{comp}(L) \in M$*, the ground literal L is interpreted as false.*

M is a partial model for a clause set N under $\prec_B$ and β if
(i) $M \models \mathrm{gnd}^{\prec_B \beta}(N)$, and
(ii) all ground literals $L\sigma$ with $L\sigma \prec_B \beta$ and $L \in C$ for some $C \in N$ are defined in M.

A ground clause C is true in a (partial) model M, denoted $M \models C$, if $C \cap M \neq \emptyset$, Conversely, a ground clause C is false in M if $\{\mathrm{comp}(L) \mid L \in C\} \subseteq M$. Otherwise, the clause is undefined in M. A ground clause set N is true in M, denoted $M \models N$ if all clauses from N are true in M.

3 Exhaustive Partial Model Exploration with SCL

In this section, we restrict model exploration to finite ground models. Hence, models are built with respect to a maximal literal β and a literal ordering $\prec_B$. For fixed β and $\prec_B$, the proposed calculus HSCL always terminates: Either by finding a refutation, or by exploring *all* partial models that are smaller than β with respect to $\prec_B$. Note that there are always only finitely many of such partial models. Of course, those (finite) models are in general not always extendable to a complete model of the clause set. However, if a complete model exists, then

there is also a corresponding partial model that is restricted to literals smaller β under $\prec_B$. Thus, this partial model is eventually explored.

If a clause set can be refuted by instantiating to ground literals smaller β, such a refutation will be found by HSCL. Additionally, we present a mechanism to *grow* the bound β in a fair way. With this growing mechanism, we show that HSCL is refutationally complete even if the original β is not sufficiently large for a refutation, because we can always grow β in a fair way until it suffices for a refutation. Thus, HSCL with Grow is refutationally complete for first-order logic.

Even in cases where no refutation exists, enumerating partial models with respect to a fixed bound yields information about the overall structure of complete models. For example, any (ground) property that holds in all enumerated partial models must be actually true. Therefore, it can be added to the current set of clauses. Properties that hold in at least some enumerated models might be a good candidate to be tested for being true in all models.

By the design of HSCL, clauses learned during the model exploration process can be re-used in later runs (e.g. a run with increased β) to speed up exploration of the original problem. To this end, HSCL will strictly distinguish between two sets of learned clauses, U and N'. During exhaustive model exploration, *excluding clauses* are learned to N', which prevent a particular model from being explored again. However, those excluding clauses are no direct consequences of N, i.e. for an excluding clause D, in general $N \not\models D$. In contrast, HSCL keeps the invariant that clauses learned to U are always direct consequences of N, i.e. $N \models U$. Thus, whenever a conflict involves any excluding clauses from N', the resulting learned clause will again be learned to N'. In contrast, if a conflict only depends on clauses from $N \cup U$, the resulting clause is a direct consequence of N and is added to U. This design allows to keep all learned clauses in U for later runs.

3.1 The HSCL Rules

The inference rules of HSCL are represented by an abstract rewrite system. They operate on a problem state, a seven-tuple $(\Gamma; N; U; \beta; N'; k; D)$ where Γ is a sequence of annotated ground literals, the *trail*; N and U are the sets of *initial* and *learned* clauses; β is a ground literal limiting the literals considered for instantiation; N' is a set of clauses that excludes all already seen partial models; k counts the number of decisions; and D is a status that is either true $\top$, false $(\bot)_R$, finished with exploration $(\bot)_\mathrm{E}$, or an annotated closure $(C \cdot \sigma)_u$, where $u \in \{\mathrm{E}, R\}$.

Literals in Γ have the form $X{:}L^r$, where $X \in \{\mathrm{D}, \mathrm{E}, \mathrm{P}\}$ is the *type* of the literal and r its justification. The justification r is a level, a closure, or the combination of a level and a closure. We often omit irrelevant parts of the justification in specific contexts. The type can either be a Decision (D), a Propagation (P) from N, or an Exclusion (E) from N'. As we do not want to explore partial models twice, decisions are no longer completely arbitrary. Instead, if a decision of a ground literal $L\sigma$ would lead to visiting a partial model again, HSCL will *exclude* this possibility by appending $\mathrm{comp}(L\sigma)$ to the trail instead. This mechanism works similar to propagation. However, instead of propagating from $N \cup U$, excluded literals are propagated from N' and are therefore treated as decision literals with respect to $N \cup U$. This mechanism is similar for conflict detection: *Regular* conflicts can be detected against clauses from $N \cup U$. In contrast, *excluded* conflicts are conflicts to a clause in N'. These two kinds of conflicts are distinguished by their respective annotation $u \in \{\mathrm{E}, R\}$.

In the trail, decided and excluded literals are annotated with a numerical level k, meaning that L is the k-th decided or excluded literal. Lastly, propagated and excluded literals are annotated with a closure that propagated the literal to become true.

A ground literal L is of *level* i with respect to a problem state of shape $(\Gamma; N; U; \beta; N'; k; D)$ if L or $\mathrm{comp}(L)$ occurs in Γ and the first decision or exclusion literal left from L ($\mathrm{comp}(L)$) in Γ, including L, is annotated with i. If there is no such decision literal then its level is zero. A ground clause D is of *level* i with respect to a problem state $(\Gamma; N; U; \beta; N'; k; D)$ if i is the maximal level of a literal in D; the level of the empty clause $\bot$ is 0. A literal L is *undefined* in Γ if neither L nor $\mathrm{comp}(L)$ occur in Γ. The initial state for a first-order clause set N is $(\epsilon; N; \emptyset; \beta; \emptyset; 0; \top)$, where β is an arbitrary but fixed literal.

The basic rules for trail building, Propagate and Decide, are left unmodified compared to the original SCL(FOL) [10] calculus, except for the difference that literals on the trail are now annotated with their respective type, i.e. whether they are decided, propagated, or excluded literals. However, in HSCL, the trail building rules are supplemented with the Exclude rule, which is similar to propagation, but propagates literals from N' instead of $N \cup U$.

Propagate $(\Gamma; N; U; \beta; N'; k; \top)$
$\Rightarrow_{\mathrm{HSCL}}$ $(\Gamma, \mathrm{P}{:}L\sigma^{(C_0 \vee L)\delta \cdot \sigma}; N; U; \beta; N'; k; \top)$
provided $C \vee L \in (N \cup U)$, $C = C_0 \vee C_1$, $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $(C \vee L)\sigma$ is ground, $(C \vee L)\sigma \prec_B \{\beta\}$, $C_0\sigma$ is false under Γ, and $L\sigma$ is undefined in Γ

The rule Propagate applies exhaustive factoring to the propagated literal with respect to the grounding substitution σ and annotates the factored clause to the propagation literal on the trail. Furthermore, note that Propagate annotates the literal on the trail with P, denoting this literal is on the trail due to a propagation.

Decide $(\Gamma; N; U; \beta; N'; k; \top)$
$\Rightarrow_{\mathrm{HSCL}}$ $(\Gamma, \mathrm{D}{:}L\sigma^{k+1}; N; U; \beta; N'; k+1; \top)$
provided $L \in C$ for a $C \in (N \cup U)$, $L\sigma$ is a ground literal undefined in Γ, and $L\sigma \prec_B \beta$

In contrast to Propagate, Decide annotates the literal with D, denoting that this literal is a decision literal.

Exclude $(\Gamma; N; U; \beta; N'; k; \top)$
$\Rightarrow_{\mathrm{HSCL}}$ $(\Gamma, \mathrm{E}{:}L\sigma^{k+1:(C_0 \vee L)\delta \cdot \sigma}; N; U; \beta; N'; k+1; \top)$
provided $C \vee L \in N'$, $C = C_0 \vee C_1$, $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $C_0\sigma$ is false under Γ, $(C \vee L)\sigma$ is ground, $(C \vee L)\sigma \prec_B \{\beta\}$, and $L\sigma$ is undefined in Γ

The rule Exclude annotates the literal with E, denoting that it is an exclusion literal. Apart from that, the Exclude rule works like Propagate, except it uses learned information from N' instead of $N \cup U$. Thus, the inferred literal is not necessarily entailed by $\Gamma \cup N$, but it prevents the generation of an already visited partial model (*stuck* state, see Definition 3 below). However, in combination with N', the literal must always be entailed by the clause set and the trail, i.e. $\Gamma \cup N \cup N' \models L\sigma$. Hence, when only considering the clause set $N \cup U$, the excluded literal will be treated as a *decision*, but when considering the clause set $N \cup U \cup N'$, the excluded literal can be treated as *propagated*. This difference will be respected in all rules below. Most rules, in their basic form, are left essentially unmodified, but have a dual version added that treats excluded literals and information from N' accordingly. In the following, the classical SCL(FOL) rules have R as a suffix, while rules that deal with exclusion have E as a suffix.

ConflictR $(\Gamma; N; U; \beta; N'; k; \top)$
$\Rightarrow_{\text{HSCL}}$ $(\Gamma; N; U; \beta; N'; k; (D \cdot \sigma)_R)$
provided $D \in (N \cup U)$, $D\sigma$ false in Γ

ConflictE $(\Gamma; N; U; \beta; N'; k; \top)$
$\Rightarrow_{\text{HSCL}}$ $(\Gamma; N; U; \beta; N'; k; (D \cdot \sigma)_{\text{E}})$
provided $D \in N'$, $D\sigma$ false in Γ

The classical rules Propagate, Decide, and ConflictR are similar to the original SCL(FOL) rules. They construct a (partial) model via the trail Γ for $N \cup U$ until a conflict, i.e., a false clause with respect to Γ is found. In HSCL, we also allow a conflict to a clause in N', meaning that all partial models that could be built with this trail have already been discovered. Thus, ConflictE signals the rewrite system that all further attempts with trail Γ should be excluded from future searches.

Clearly, these two kinds of conflicts need to be separated. A conflict is annotated $(D \cdot \sigma)_R$ if it was a regular conflict to a clause from $N \cup U$. In contrast, $(D \cdot \sigma)_{\text{E}}$ denotes a conflict to a clause from N', i.e., the current state can only produce partial models that were already visited and can therefore be excluded.

If a conflict is found, it is resolved by the conflict resolution rules below. Before any conflict resolution step, we assume that the respective clauses are renamed such that they do not share any variables and that the grounding substitutions of closures are adjusted accordingly.

Skip $(\Gamma, X{:}L; N; U; \beta; N'; k; (D \cdot \sigma)_u)$
$\Rightarrow_{\text{HSCL}}$ $(\Gamma; N; U; \beta; N'; k - i; (D \cdot \sigma)_u)$
provided $\text{comp}(L)$ does not occur in $D\sigma$, and if $X \in \{D, E\}$, i.e. L is a decision or exclusion literal, then $i = 1$, else $i = 0$

Factorize $(\Gamma; N; U; \beta; N'; k; ((D \vee L \vee L') \cdot \sigma)_u)$
$\Rightarrow_{\text{HSCL}}$ $(\Gamma; N; U; \beta; N'; k; ((D \vee L)\eta \cdot \sigma)_u)$
provided $L\sigma = L'\sigma$, $\eta = \text{mgu}(L, L')$

ResolveR $(\Gamma, \text{P}{:}L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; N'; k; ((D \vee L') \cdot \sigma)_R)$
$\Rightarrow_{\text{HSCL}}$ $(\Gamma, \text{P}{:}L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; N'; k; ((D \vee C)\eta \cdot \sigma\delta)_R)$
provided $L\delta = \text{comp}(L'\sigma)$, $\eta = \text{mgu}(L, \text{comp}(L'))$

Note that Skip, Factorize and ResolveR strongly resemble the original SCL(FOL) rules. In particular, ResolveR does not remove the literal $L\delta$ from the trail. This is needed if the clause $D\sigma$ contains further literals complementary of $L\delta$ that have not been factorized. Note that ResolveR resolves a conflict with respect to $N \cup U$. Hence, it does only allow to resolve with propagated literals of type P, as they are propagations from $N \cup U$. In contrast, it does not allow resolving with any excluded literal, as they are propagations from N'. Hence, excluded literals must be treated like decision when resolving regular conflicts.

ResolveE $(\Gamma, X{:}L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; N'; k; ((D \vee L') \cdot \sigma)_{\text{E}})$
$\Rightarrow_{\text{HSCL}}$ $(\Gamma, X{:}L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; N'; k; ((D \vee C)\eta \cdot \sigma\delta)_{\text{E}})$
provided $X \in \{\text{E}, \text{P}\}$, $L\delta = \text{comp}(L'\sigma)$, $\eta = \text{mgu}(L, \text{comp}(L'))$

In contrast to ResolveR, the rule ResolveE resolves a conflict with respect to $N \cup U \cup N'$. Hence, both regular propagations from $N \cup U$ (literals on the trail of type P), and excluded literals from N' (literals of type E) can be resolved with during applications of ResolveE.

BacktrackR $(\Gamma_0, K, \Gamma_1, X : \mathrm{comp}(L\sigma)^k; N; U; \beta; N'; k; ((D \vee L) \cdot \sigma)_R)$
$\Rightarrow_{\mathrm{HSCL}}$ $(\Gamma_0; N; U \cup \{D \vee L\}; \beta; N'; j - i; \top)$
provided $X \in \{\mathrm{E}, \mathrm{D}\}$ and $D\sigma$ is of level $i' < k$, and Γ_0, K is the minimal trail subsequence such that there is a grounding substitution τ with $(D \vee L)\tau$ is false in Γ_0, K but not in Γ_0, the literal K is of level j, if K is a decision or an exclusion literal then $i = 1$, otherwise $i = 0$

BacktrackE $(\Gamma_0, K, \Gamma_1, D : \mathrm{comp}(L\sigma)^k; N; U; \beta; N'; k; ((D \vee L) \cdot \sigma)_{\mathrm{E}})$
$\Rightarrow_{\mathrm{HSCL}}$ $(\Gamma_0; N; U; \beta; N' \cup \{D \vee L\}; j - i; \top)$
provided $D\sigma$ is of level $i' < k$, and Γ_0, K is the minimal trail subsequence such that there is a grounding substitution τ with $(D \vee L)\tau$ is false in Γ_0, K but not in Γ_0, the literal K is of level j, if K is a decision or an exclusion literal then $i = 1$, otherwise $i = 0$

While BacktrackR learns to the clause set U, BacktrackE learns clauses to N'. Here, BacktrackR can also jump back to excluded literals since they are treated like decisions w.r.t. $N \cup U$. In contrast, excluded literals are propagations for conflicts with $N \cup U \cup N'$ and, thus, cannot be backtracked to. The clause $D \vee L$ added by the rule BacktrackR to U is called a *learned clause*. Similarly, clauses added by BacktrackE to N' are called *excluding clauses*.

Please note that the corner case $j + 1 = k$ and $\tau = \sigma$ is also part of both backtrack rules. The rules backtrack to the minimal trail where the clause $D \vee L$ propagates. Also, note that the existence of the trail prefix Γ_0, K is guaranteed if $(D \vee L)\sigma \neq \bot$ and all other preconditions of the rule are met. Then, $(D \vee L)\sigma$ is false under $\Gamma_0, K, \Gamma_1, \mathrm{comp}(L\sigma)$ by soundness (see Definition 5). However, $(D \vee L)\sigma$ must be undefined and hence not false under the empty trail. Thus, there must be an intermediate literal K on the trail where the demanded property holds.

Definition 3 (Stuck State)**.** *A state* $(M; N; U; \beta; N'; k; D)$ *is called* stuck *if* $D \neq (\bot \cdot \sigma)_u$ *and none of the above rules are applicable.*

In classical SCL(FOL), no further rule is applicable to a stuck state. However, in HSCL, the following rule will allow further exploration from a stuck state:

Unstuck $(\Gamma; N; U; \beta; N'; k; \top) \Rightarrow_{\mathrm{HSCL}} (\epsilon; N; U; \beta; N' \cup \{C\}; 0; \top)$
provided $(\Gamma; N; U; \beta; N'; k; \top)$ is a stuck state, $C = \bigvee_{L_i \in \mathrm{decision}(\Gamma)} \mathrm{comp}(L_i)$

In this rule, the function decision(Γ) collects all decided literals D:L_k that have been introduced by applications of the rule Decide. If no such literal exists in Γ, then $C = \bot$. Currently, this rule clears the trail Γ. However, this is not necessary. An implementation might choose to keep the trail. In this case, the next application of a rule is ConflictE, which detects a conflict to the new clause, which can then be resolved. In general, an implementation can even jump to any prefix of the current trail Γ.

The Unstuck rule allows us to explore all stuck states. Whenever a stuck state is found, the trail is reset, and this particular stuck state will be prevented from being explored again by adding the clause consisting of the complement of all decisions to the exclusion set N'. Notably, stuck states directly correspond to partial models bound by $\prec_B \beta$, see Definition 18. Thus, exploring stuck states is a way to get insights on the literal structure of partial models. In the following, we will construct regularity rules such that a complete run will eventually explore all

stuck states. Thus, by exploring all stuck states, this run also enumerates all partial models w.r.t. $\prec_B$ and β, as shown in Theorem 19. A HSCL run is *finished* with the exploration of $\mathrm{gnd}^{\prec_B \beta}(N)$ if it is in the state $(\Gamma; N; U; \beta; N'; k; (\bot)_{\mathrm{E}})$. In contrast, a HSCL run has detected unsatisfiability of the clause set N if is in a state $(\Gamma; N; U; \beta; N'; k; (\bot)_R)$

Example 4 (HSCL rules and Learning after Unstuck)**.** *A first simple example showing the application of the calculus rules is as follows. Consider* $N = \{C_1 = P(x) \vee Q(x), C_2 = \neg P(x) \vee \neg Q(x), C_3 = P(x) \vee \neg Q(x)\}$*. Let* σ *denote the substitution* $\{x \mapsto a\}$*. Choose* β *and* $\prec_B$ *in a way that only* $\{P(a), \neg P(a), Q(a), \neg Q(a)\} \prec_B \{\beta\}$*. Then, the following HSCL run explores all partial models for* N *with respect to* $\prec_B$ *and* β*:*

$$
\begin{aligned}
& (\varepsilon; N; \emptyset; \beta; \emptyset; 0; \top) \\
\Rightarrow^{Decide}_{HSCL} \quad & (D{:}P(a)^1; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Propagate}_{HSCL} \quad & (D{:}P(a)^1, P{:}\neg Q(a)^{C_2 \cdot \sigma}; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Unstuck}_{HSCL} \quad & (\varepsilon; N; \emptyset; \beta; N' = \{\neg P(a)\}; 0; \top) \\
\Rightarrow^{Exclude}_{HSCL} \quad & (E{:}\neg P(a)^{1:\neg P(a)}; N; \emptyset; \beta; N'; 1; \top) \\
\Rightarrow^{Propagate}_{HSCL} \quad & (E{:}\neg P(a)^{1:\neg P(a)}, P{:}Q(a)^{C_1 \cdot \sigma}; N; \emptyset; \beta; N'; 1; \top) \\
\Rightarrow^{ConflictR}_{HSCL} \quad & (E{:}\neg P(a)^{1:\neg P(a)}, P{:}Q(a)^{C_1 \cdot \sigma}; N; \emptyset; \beta; N'; 1; (C_3 \cdot \sigma)_R) \\
\Rightarrow^{ResolveR}_{HSCL} \quad & (E{:}\neg P(a)^{1:\neg P(a)}, P{:}Q(a)^{C_1 \cdot \sigma}; N; \emptyset; \beta; N'; 1; (P(x) \vee P(x) \cdot \sigma)_R) \\
\Rightarrow^{Factorize}_{HSCL} \quad & (E{:}\neg P(a)^{1:\neg P(a)}, P{:}Q(a)^{C_1 \cdot \sigma}; N; \emptyset; \beta; N'; 1; (P(x) \cdot \sigma)_R) \\
\Rightarrow^{Skip}_{HSCL} \quad & (E{:}\neg P(a)^{1:\neg P(a)}; N; \emptyset; \beta; N'; 1; (P(x) \cdot \sigma)_R) \\
\Rightarrow^{BacktrackR}_{HSCL} \quad & (\varepsilon; N; \{P(x)\}; \beta; N'; 0; \top) \\
\Rightarrow^{Propagate}_{HSCL} \quad & (P{:}P(a)^{P(x) \cdot \sigma}; N; \{P(x)\}; \beta; N'; 0; \top) \\
\Rightarrow^{ConflictE}_{HSCL} \quad & (P{:}P(a)^{P(x) \cdot \sigma}; N; \{P(x)\}; \beta; N'; 0; (\neg P(a))_E) \\
\Rightarrow^{ResolveE}_{HSCL} \quad & (P{:}P(a)^{P(x) \cdot \sigma}; N; \{P(x)\}; \beta; N'; 0; (\bot)_E)
\end{aligned}
$$

In this example, there is only one partial model $\{P(a), \neg Q(a)\}$*. Hence, Unstuck is only applied once in the overall HSCL run. Note that after the first use of Unstuck, a regular conflict is detected and resolved. In particular, our calculus learns the new clause* $P(x)$ *to* U *after the application of BacktrackR. This clause can be kept for future searches and simplifies the clause set. Note that classical SCL without Unstuck would not have learned the clause in a comparable run. The final state of this run is* $(\varepsilon; N; \{P(x)\}; \beta; N'; 0; (\bot)_E)$*, meaning that HSCL is finished with exploring all partial models w.r.t.* $\prec_B$ *and* β*.*

A more complex example will be presented at the end of this section.

3.2 HSCL Properties

In this section, we will prove the key properties of HSCL. This starts with soundness (Theorem 7), where we will prove that the following invariants are preserved over all HSCL runs:

Definition 5 (Sound States)**.** *A state* $(\Gamma; N; U; \beta; N'; k; D)$ *is* sound *if the following conditions hold:*

1. Γ *is a consistent sequence of annotated ground literals,*
2. *for each decomposition* $\Gamma = \Gamma_1, P{:}L\sigma^{C \vee L \cdot \sigma}, \Gamma_2$ *we have that* $C\sigma$ *is false under* Γ_1 *and* $L\sigma$ *is undefined under* Γ_1*,* $N \cup U \models C \vee L$

3. *for each decomposition* $\Gamma = \Gamma_1, E{:}L\sigma^{k:C \vee L \cdot \sigma}, \Gamma_2$ *we have that* $C\sigma$ *is false under* Γ_1 *and* $L\sigma$ *is undefined under* Γ_1, $N \cup N' \models C \vee L$

4. *for each decomposition* $\Gamma = \Gamma_1, X{:}L, \Gamma_2$ *we have that* L *is undefined in* Γ_1,

5. $N \models U$,

6. *if* $D = (C \cdot \sigma)_R$ *then* $C\sigma$ *is false under* Γ *and* $N \models C$. *In particular,* $\mathrm{gnd}^{\prec_B \beta}(N) \models C\sigma$.

7. *if* $D = (C \cdot \sigma)_E$ *then* $C\sigma$ *is false under* Γ *and* $N \cup N' \models C$. *In particular,* $\mathrm{gnd}^{\prec_B \beta}(N \cup N') \models C\sigma$.

8. *for any* $L\sigma \in \Gamma$ *we have* $L\sigma \prec_B \beta$ *and there is a* $C \in (N \cup U)$ *such that* $L \in C$.

Lemma 6 (Initial state soundness)**.** *The initial state* $(\epsilon; N; \emptyset; \beta; \emptyset; 0; \top)$ *is sound.*

Proof. Criteria 1–4 and 8 are trivially satisfied by $\Gamma = \epsilon$. Furthermore, $N \models \emptyset$, fulfilling criterion 5. Lastly, criteria 6 and 7 are trivially fulfilled for $D = \top$. □

Theorem 7 (Soundness of the HSCL rules)**.** *All HSCL rules preserve sound states.*

Corollary 8. *Assume a HSCL state* $(\Gamma; N; U; \beta; N'; k; D)$ *resulting from a HSCL run. Then,* $(\Gamma; N; U; \beta; N'; k; D)$ *is sound.*

Proof. Follows with induction over the size of the run. The base case is handled by Lemma 6, the induction step is contained in Theorem 7. □

While all runs of HSCL preserve the soundness invariants, another key property of HSCL is *non-redundant learning*, which we will prove in Theorem 13. However, to guarantee non-redundant learning, note that we must further restrict the application of HSCL rules. The key idea of non-redundant learning is to restrict HSCL runs in a fashion that during conflict resolution, the rule ResolveR or ResolveE is invoked *at least once*. In Definition 9 and 10, we define *reasonable* and *regular* runs, which provide this property. Overall, in a regular run, it is guaranteed that during conflict resolution, we resolve at least once with ResolveR or ResolveE (Lemma 11).

Next, we define a suitable, dynamic ordering (see Definition 12). Under this ordering, a clause will become smaller after any applied resolution step. Hence, in a regular run, it is guaranteed that one resolution step takes place, which will produce a smaller clause w.r.t. our dynamic ordering. Moreover, we prove that an eventually learned clause in a regular run is not only smaller, but must even be non-redundant to all previously seen clauses with respect to this dynamic ordering (Theorem 13). Without the restriction to regular runs, it is possible to learn redundant clauses, as demonstrated in the scenario of Example 14. Hence, this motivates a restriction to regular runs, as regularity is sufficient to guarantee non-redundant learning.

Definition 9 (Reasonable Runs)**.** *A sequence of HSCL rule applications is called a* reasonable run *if it meets the following two criteria:*

- *The rule Decide does not enable an immediate application of the rule ConflictR or ConflictE.*
- *The rule Exclude does not enable an immediate application of the rule ConflictR.*

Definition 10 (Regular HSCL Runs)**.** *A sequence of HSCL rule applications is called a* regular run *if it is a reasonable run and ConflictR and ConflictE always have priority over every other rule.*

Lemma 11 (Regular Conflict Resolution in HSCL)**.** *Consider HSCL in the conflict state* $(\Gamma, X{:}L; N; U; \beta; N'; k; (D)_u)$*. In a regular run, during conflict resolution, at least the rightmost literal* L *is resolved with.*

Definition 12 (State Induced Ordering)**.** *Let* $(L_1, L_2, \ldots, L_n; N; U; \beta; N'; k; D)$ *be a sound HSCL state. The trail induces a total well-founded strict order on the defined literals by*

$$L_1 \prec_\Gamma \operatorname{comp}(L_1) \prec_\Gamma L_2 \prec_\Gamma \operatorname{comp}(L_2) \prec_\Gamma \cdots \prec_\Gamma L_n \prec_\Gamma \operatorname{comp}(L_n)$$

We extend $\prec_\Gamma$ *to a strict total order on all literals where all undefined literals are larger than* $\operatorname{comp}(L_n)$*. We also extend* $\prec_\Gamma$ *to a strict total order on ground clauses by multiset extension and also on multisets of ground clauses and overload* $\prec_\Gamma$ *for all these cases. With* $\preceq_\Gamma$ *we denote the reflexive closure of* $\prec_\Gamma$*.*

Theorem 13 (Non-redundant learning in HSCL)**.** *Let* $(\Gamma; N; U; \beta; N'; k; (C_0 \cdot \sigma_0)_u)$ *be the state after an application of ConflictR (resp. ConflictE) in a regular run and let* C *be the clause learned at the end of the conflict resolution, then* C *is not redundant with respect to* $N \cup U$ *(resp.* $N \cup N' \cup U$*) and* $\prec_\Gamma$*.*

Of course, in a regular run, the ordering of literals on the trail will change, i.e., the ordering of Definition 12 will change as well. Thus, the non-redundancy property of Theorem 13 reflects the situation at the time of creation of the learned clause. A non-redundancy property holding for an overall run must be invariant against changes on the ordering. The ordering of Definition 12 includes the subset ordering $\prec_\subseteq$ that is invariant against changes on the overall ordering. This means that our dynamic ordering entails non-redundancy criteria based on (strict) subset relations including, e.g., subsumption. From an implementation perspective, this means that learned clauses need not to be tested for forward redundancy. Current resolution or superposition based provers spend a reasonable portion of their time in testing forward redundancy of newly generated clauses. In addition, also tests for backward reduction can be simplified knowing that learned clauses are not redundant. For example, consider a run that learned the clauses $C_1, \ldots, C_n$ in chronological order, where each C_i is non-redundant with respect to $\{C_1, \ldots, C_{i-1}\}$. Now, when checking C_i for backward subsumption, it is sufficient to check if any of $C_{i+1}, \ldots, C_n$ subsumes C_i.

Recall that Theorem 13 only holds for regular runs. The following example shows a non-regular run which can learn a redundant clause.

Example 14 (Learning redundant clauses without regularity)**.** *Consider the following clause set*

$$N = \{P(x) \vee Q(x), \quad P(x) \vee \neg Q(x), \quad \neg P(x) \vee Q(x), \quad \neg P(x) \vee \neg Q(x)\}$$

and $\beta, \prec_B$ *chosen such that exactly* $\{P(a), \neg P(a), Q(a), \neg Q(a)\} \prec_B \{\beta\}$ *Now, consider the following fragment of a* non-regular *HSCL run which learns the redundant clause* $\neg P(x) \vee \neg Q(x)$*:*

$$\begin{aligned}
& (\varepsilon; N; \emptyset; \beta; \emptyset; 0; \top) \\
\Rightarrow^{Decide}_{HSCL} \quad & (D{:}P(a)^1; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Decide}_{HSCL} \quad & (D{:}P(a)^1 D{:}Q(a)^2; N; \emptyset; \beta; \emptyset; 2; \top) \\
\Rightarrow^{ConflictR}_{HSCL} \quad & (D{:}P(a)^1 D{:}Q(a)^2; N; \emptyset; \beta; \emptyset; 1; ((\neg P(x) \vee \neg Q(x)) \cdot \{x \mapsto a\})_R) \\
\Rightarrow^{BacktrackR}_{HSCL} \quad & (D{:}P(a)^1; N; \{\neg P(x) \vee \neg Q(x)\}; \beta; \emptyset; 1; \top)
\end{aligned}$$

The learned clause is trivially redundant, as it is already contained in N*.*

Hence, to guarantee non-redundant learning, we restrict our view to regular HSCL runs. Next, for regular runs, we prove correct termination of HSCL. Essentially, Lemma 15 proves that HSCL without Unstuck can end up in three possible states: Either by finding a refutation (then $D = (\bot)_R$), or by finding that there are no more new partial models that are not yet excluded in N' (then $D = (\bot)_\mathrm{E}$), or by ending up in a stuck state itself. Of course, HSCL with Unstuck cannot end up in a stuck state, so it must end up in any of the first two options, Corollary 16.

Lemma 15 (Correct Termination without Unstuck)**.** *If in a regular run no rule except Unstuck is applicable to an HSCL state* $(\Gamma; N; U; \beta; N'; k; D)$*, then either* $D = (\bot)_R$*, or* $D = (\bot)_E$*, or* $D = \top$ *and* $\Gamma \models \mathrm{gnd}^{\prec_B \beta}(N)$*.*

Corollary 16 (Correct Termination of HSCL)**.** *If in a regular run no rules are applicable to a state* $(\Gamma; N; U; \beta; N'; k; D)$*, then either* $D = (\bot \cdot \sigma)_R$ *and* N *is unsatisfiable, or* $D = (\bot)_E$ *and* $N' \neq \emptyset$ *and Unstuck was applied at least once.*

Proof. We instantiate Lemma 15. If $D = (\bot)_R$, by soundness it must hold that $N \models \bot$. Hence, N must be unsatisfiable. In the case $D = (\bot)_\mathrm{E}$, note that $N' \neq \emptyset$, since a conflict $(C)_\mathrm{E}$ can only be produced by ConflictE. In particular, there must have been at least one application of Unstuck which led to a non-empty N'. Otherwise, if $D = \top$, Unstuck can always be applied to our state by definition. □

Theorem 17 (Termination of HSCL)**.** *All regular HSCL runs terminate.*

Proof. In Theorem 15, we proved that all regular runs which do not use Unstuck terminate. Thus, it is left to show that Unstuck cannot be used infinitely often. Note that every regular use of Unstuck on a state $(\Gamma; N; U; \beta; N'; k; D)$ adds a clause C to N'. However, by Theorem 13, C is not redundant to $N \cup U \cup N'$ under $\prec_\subseteq$, which is well-founded. Due to the restriction of all learned clauses to be smaller than $\{\beta\}$, the number of non-redundant ground clauses is finite. Thus, Unstuck cannot be applied infinitely often, and all regular HSCL runs must terminate. □

Lemma 15 forms the basis of the following observation captured in Definition 18. It states that if no rule except Unstuck is applicable to a state and $D = \top$, then already $\Gamma \models \mathrm{gnd}^{\prec_B \beta}(N)$. This means that in a stuck state, our trail already forms a partial model for $\mathrm{gnd}^{\prec_B \beta}(N)$. Finally, we are going to prove in Theorem 19 that HSCL indeed exhaustively enumerates all partial models. Essentially, we prove that a stuck state corresponding to *any* partial model is visited during a complete run of HSCL.

Definition 18 (Stuck states correspond to partial models)**.** *Let* M *be a partial model for* N *under* $\prec_B$ *and* β*, i.e.,* $M \models \mathrm{gnd}^{\prec_B \beta}(N)$ *and all ground literals* $L\sigma$ *with* $L\sigma \prec_B \{\beta\}$ *and* $L \in C$ *for a* $C \in N$ *are defined in* M*. We call a stuck state* $(\Gamma; N; U; \beta; N'; k; D)$ *corresponding to* M *if* $\Gamma \models M$*.*

Theorem 19 (Exhaustive stuck state exploration)**.** *Consider a HSCL run that ends in the final state* $(\Gamma; N; U; \beta; N'; k; (\bot)_E)$*. For all partial models* M *of* N *under* $\prec_B$ *and* β*, a stuck state corresponding to* M *is eventually explored in such a run.*

Corollary 20. *If a clause set* N *is satisfiable with a model* M*, then the partial model* $M' \subseteq M$ *for* N *under* $\prec_B$ *and* β *is explored during a regular HSCL run.*

Proof. If M is a model for N, then M' is a model for $\mathrm{gnd}^{\prec_B \beta}(N)$. With Theorem 19, as all partial models for N under $\prec_B$ and β are explored, M' must be explored as well. □

Corollary 20 stands in contrast to classical SCL(FOL), where only one partial model with respect to the bounding $\prec_B$ and β is found. Hence, it can be the case that a non-extendable model is found during an SCL run. In contrast, as our HSCL run exhaustively explores *all* models, all extendable models (if any) will be found as well.

Example 21 (Enumerating multiple models with HSCL)**.** *For a more complex example, consider*

$$N = \left\{ \begin{array}{lll} C_1 & = \neg P(x) \vee Q(x) & C_2 \quad = \neg Q(x) \vee R(x) \\ C_3 & = \neg R(x) \vee P(x) \vee Q(x) & C_4 \quad = R(a) \end{array} \right\}$$

and $\beta, \prec_B$ chosen such that exactly

$$\{P(a), \neg P(a), Q(a), \neg Q(a), R(a), \neg R(a)\} \prec_B \{\beta\}$$

Furthermore, let σ denote the substitution $\{x \mapsto a\}$. For example, a regular HSCL run could explore all partial models in the following way:

$$\begin{array}{ll}
 & (\varepsilon; N; \emptyset; \beta; \emptyset; 0; \top) \\
\Rightarrow^{Propagate}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}; N; \emptyset; \beta; \emptyset; 0; \top) \\
\Rightarrow^{Decide}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}, D{:}\neg Q(a)^1; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Propagate}_{HSCL} & (\Gamma = P{:}R(a)^{C_4 \cdot \{\}}, D{:}\neg Q(a)^1, P{:}P(a)^{C_3 \cdot \sigma}; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{ConflictR}_{HSCL} & (\Gamma; N; \emptyset; \beta; \emptyset; 1; (\neg P(x) \vee Q(x) \cdot \sigma)_R) \\
\Rightarrow^{ResolveR}_{HSCL} & (\Gamma; N; \emptyset; \beta; \emptyset; 1; (\neg R(x) \vee Q(x) \vee Q(x) \cdot \sigma)_R) \\
\Rightarrow^{Factorize}_{HSCL} & (\Gamma; N; \emptyset; \beta; \emptyset; 1; (\neg R(x) \vee Q(x) \cdot \sigma)_R) \\
\Rightarrow^{Skip}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}, D{:}\neg Q(a)^1; N; \emptyset; \beta; \emptyset; 1; (\neg R(x) \vee Q(x) \cdot \sigma))_R \\
\Rightarrow^{BacktrackR}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}; N; U = \{C_5 = \neg R(x) \vee Q(x)\}; \beta; \emptyset; 0; \top) \\
\Rightarrow^{Decide}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}, D{:}P(a)^1; N; U; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Propagate}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}, D{:}P(a)^1, P{:}Q(a)^{C_5 \cdot \sigma}; N; U; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Unstuck}_{HSCL} & (\varepsilon; N; U; \beta; N' = \{\neg P(a)\}; 0; \top) \\
\Rightarrow^{Propagate}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}; N; U; \beta; N'; 0; \top) \\
\Rightarrow^{Propagate}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}, P{:}Q(a)^{C_5 \cdot \sigma}; N; U; \beta; N'; 0; \top) \\
\Rightarrow^{Exclude}_{HSCL} & (P{:}R(a)^{C_4 \cdot \{\}}, P{:}Q(a)^{C_5 \cdot \sigma}, E{:}\neg P(a)^{1:\neg P(a) \cdot \{\}}; N; U; \beta; N'; 1; \top) \\
\Rightarrow^{Unstuck}_{HSCL} & (\varepsilon; N; U; \beta; N' \cup \{\bot\}; 0; \top) \\
\Rightarrow^{ConflictE}_{HSCL} & (\varepsilon; N; U; \beta; N' \cup \{\bot\}; 0; (\bot)_E)
\end{array}$$

Just before any use of Unstuck, the trail forms a partial model under $\prec_B$ and β. In this example, the partial models are $\{P(a), Q(a), R(a)\}$ and $\{\neg P(a), Q(a), R(a)\}$.

3.3 Increasing bounds

The previous rules provide a way to enumerate all bounded partial models with respect to a fixed literal β. In addition, HSCL can be turned into a semi-decision procedure for first-order logic. For this, in general, β must be increased during a run. This is achieved by the Grow rule:

Grow $\quad (\Gamma; N; U; \beta; N'; k; \top) \Rightarrow_{\text{SCL}} (\Gamma; N; U; \beta'; \emptyset; k; \top)$
provided $\beta \prec_B \beta'$

Moreover, in classical CDCL, a rule Restart that clears the trail proved beneficial. Thus, in a similar fashion, we can add the following rule to HSCL:

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

Adding both rules does not change soundness and correct termination of the calculus. This is captured by the following two corollaries, which are straight-forward to prove. In particular, note that Corollary 23 is essentially just a restated version of Corollary 16, as we exclude both Grow and Restart explicitly.

Corollary 22 (Soundness of HSCL with Grow and Restart)**.** *The rules Grow and Restart preserve the soundness properties from Definition 5.*

Corollary 23 (Correct termination of HSCL with Grow and Restart)**.** *If in a run no rule except Grow and Restart is applicable to a state* $(\Gamma; N; U; \beta; N'; k; D)$*, then either* $D = (\bot \cdot \sigma)_R$ *and* N *is unsatisfiable, or* $D = (\bot)_E$ *and* N *has at least one partial model w.r.t.* $\prec_B$ *and* β*.*

Soundness and correct termination are preserved in a straight-forward way when adding the Grow and Restart rule. In contrast, HSCL with those added rules does not always terminate. For example, it is possible to restart infinitely often without making progress. Thus, it is necessary to restrict the usage of both Grow and Restart for termination. To this end, we define a *fair* run as follows:

Definition 24 (Fair runs of HSCL)**.** *A HSCL run is called fair if*

- *Restart is applied only finitely often*
- *Grow is applied only in stuck states or to a state* $(\Gamma; N; U; \beta; N'; k; (\bot)_E)$*.*

Still, in general, a fair run of HSCL with Grow will not always terminate in case of infinite models. For example, in a clause set $N = \{P(a), \neg P(x) \vee P(f(x))\}$, all ground models are infinite. Thus, a HSCL run can infinitely often reach a stuck state and apply the Grow rule to it. However, HSCL with this extension is *refutationally complete*. Hence, for an unsatisfiable clause set N, HSCL will eventually terminate and find a refutation. Overall, this turns HSCL into a semi-decision procedure for unsatisfiability of first-order logic.

To prove refutational completeness, we first prove a more general statement in Theorem 25. This theorem guarantees that HSCL cannot deal with a non-extendable partial model infinitely long. As a corollary, if the input clause set is unsatisfiable, the empty partial model is not extendable. Hence, HSCL will terminate after finitely many steps in this case, see Theorem 26.

Theorem 25 (Eliminating non-extendable stuck states)**.** *Consider a regular HSCL run in a stuck state* $(\Gamma; N; U; \beta; N'; k; \top)$*. Then, either* Γ *is extendable to a model for* N*, or after finitely many applications of Grow, there is no more stuck state* $(\Gamma'; N; U; \beta; N'; k; \top)$ *reachable in this HSCL run, where* $\Gamma \subseteq \Gamma'$*.*

Theorem 26 (Refutational completeness of HSCL with Grow)**.** *Consider an unsatisfiable clause set* N*. A regular and fair HSCL run with Grow will terminate after finitely many steps in the finished state* $(\Gamma; N; U; \beta; N'; k; (\bot)_R)$*.*

Proof. By Theorem 17, a regular HSCL run without Grow or Restart terminates. By fairness, Restart can only be applied finitely often. It remains to show that Grow cannot be applied infinitely often.

Let R be a fair and regular run of HSCL on N with starting state $(\varepsilon; N; \emptyset; \beta; \emptyset; 0; \top)$. Then, for β_0 being a new, smallest literal in $\prec_B$, the following prefix of R

$$(\varepsilon; N; \emptyset; \beta_0; \emptyset; 0; \top) \Rightarrow_{\text{HSCL}}^{\text{Grow}} (\varepsilon; N; \emptyset; \beta; \emptyset; 0; \top)$$

forms again a regular and fair run. By Theorem 25, after finitely many applications of Grow there is no more stuck state $\Gamma' \supseteq \varepsilon$, i.e. there is overall no more stuck state. Hence, in a fair run, Grow cannot be applied anymore. Note that the last application of Grow produced a state where $N' = \emptyset$. As there is no more stuck state, Unstuck cannot be applied. Thus, $N' = \emptyset$ for the remainder of the run. Then, by Corollary 16, HSCL cannot end up in $(\bot)_{\text{E}}$, so it must end up in $(\bot)_R$. □

Theorem 27 (Learning from non-extendable models)**.** *Consider a regular HSCL run in a stuck state $(\Gamma; N; U; \beta; N'; k; \top)$, where Γ is not extendable to a model. Then, after finitely many applications of Grow, HSCL can either find a refutation or learn a clause C to U that prevents Γ from being explored again, i.e. $\Gamma \not\models C\sigma$.*

HSCL is guaranteed to learn such a clause by following a regular, fair strategy that does neither apply Restart nor Unstuck after encountering the stuck state.

In contrast to Theorem 25, the latter Theorem 27 actually provides an *explicit* way to learn a single clause C that prevents this stuck state from being explored again. In contrast to the Unstuck rule, which learns a blocking clause to N', this learned clause C is actually a consequence of N. Hence, it can be learned to U, keeping the invariant $N \models U$. In particular, this clause can be kept for future searches with extended bounds, as U is not cleared by the rule Grow.

Theorem 27 guarantees to learn such a clause if HSCL follows the strategy of not applying Restart or Unstuck after the stuck state is encountered. However, note that there are multiple different ways to learn such a clause. In particular, using Restart or Unstuck does not make learning such a clause impossible. Instead, even after a single use of Restart or Unstuck, HSCL is simply no longer *guaranteed* to learn such a clause. The reason is that HSCL might use Restart and Unstuck to avoid the non-extendable model in the future. It is important to note that this does not stop HSCL from being refutationally complete (see Theorem 26). Moreover, HSCL is also guaranteed to learn the preventing clause if it always exhaustively explores all partial models for a given β, i.e. it does not apply Grow in a stuck state but only in a state where $D = (\bot)_{\text{E}}$.

Example 28 (Learning from non-extendable models)**.** *To demonstrate HSCL with Grow, and in particular learning from non-extendable models (Theorem 27), consider a HSCL run on the following clause set:*

$$N = \left\{ \begin{array}{ll} (1) & \neg P(x) \vee P(g(x)) \\ (2) & \neg P(g(g(a))) \end{array} \right\}$$

First, we choose $\prec_B$ as a KBO, where all symbols have weight one and with precedence $a \prec g \prec P$. In the beginning of the run, we set $\beta = P(g(g(a)))$. We use the notation $g^n(x)$ as a shorthand for n-time application of g to x. Then, a regular and fair HSCL run on N could look as follows:

$$
\begin{array}{ll}
 & (\varepsilon; N; \emptyset; \beta; \emptyset; 0; \top) \\
\Rightarrow^{Decide}_{HSCL} & (D{:}P(a)^1; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{PropagateR}_{HSCL} & (D{:}P(a)^1, P{:}P(g(a))^{(1)\cdot\{x\mapsto a\}}; N; \emptyset; \beta; \emptyset; 1; \top) \\
\Rightarrow^{Grow}_{HSCL} & (D{:}P(a)^1, P{:}P(g(a))^{(1)\cdot\{x\mapsto a\}}; N; \emptyset; \beta' = P(g^3(a)); \emptyset; 1; \top) \\
\Rightarrow^{PropagateR}_{HSCL} & (\Gamma = D{:}P(a)^1, P{:}P(g(a))^{(1)\cdot\{x\mapsto a\}}, P{:}P(g^2(a))^{(1)\cdot\{x\mapsto g(a)\}}; N; \emptyset; \beta'; \emptyset; 1; \top) \\
\Rightarrow^{ConflictR}_{HSCL} & (\Gamma; N; \emptyset; \beta'; \emptyset; 1; ((2)\cdot\{\})_R) \\
\Rightarrow^{ResolveR}_{HSCL} & (\Gamma; N; \emptyset; \beta'; \emptyset; 1; (\neg P(g(a))\cdot\{\})_R) \\
\Rightarrow^{Skip}_{HSCL} & (D{:}P(a)^1, P{:}P(g(a))^{(1)\cdot\{x\mapsto a\}}; N; \emptyset; \beta'; \emptyset; 1; (\neg P(g(a))\cdot\{\})_R) \\
\Rightarrow^{ResolveR}_{HSCL} & (D{:}P(a)^1, P{:}P(g(a))^{(1)\cdot\{x\mapsto a\}}; N; \emptyset; \beta'; \emptyset; 1; (\neg P(a)\cdot\{\})_R) \\
\Rightarrow^{Skip}_{HSCL} & (D{:}P(a)^1; N; \emptyset; \beta'; \emptyset; 1; (\neg P(a)\cdot\{\})_R) \\
\Rightarrow^{BacktrackR}_{HSCL} & (\varepsilon; N; U' = \{\neg P(a)\}; \beta'; \emptyset; 0; \top) \\
\Rightarrow^{PropagateR}_{HSCL} & (P{:}\neg P(a)^{\neg P(a)\cdot\{\}}; N; U'; \beta'; \emptyset; 0; \top) \\
\Rightarrow^{PropagateR}_{HSCL} & (P{:}\neg P(a)^{\neg P(a)\cdot\{\}}, P{:}\neg P(g^2(a))^{(2)\cdot\{\}}; N; U'; \beta'; \emptyset; 0; \top) \\
\Rightarrow^{PropagateR}_{HSCL} & (P{:}\neg P(a)^{\neg P(a)\cdot\{\}}, P{:}\neg P(g^2(a))^{(2)\cdot\{\}}, P{:}\neg P(g(a))^{(1)\cdot\{x\mapsto g(a)\}}; N; U'; \beta'; \emptyset; 0; \top) \\
\Rightarrow^{Unstuck}_{HSCL} & (\varepsilon; N; U'; \beta'; \emptyset; 0; (\bot)_E)
\end{array}
$$

Note that N has two partial models with respect to $\prec_B$ and $\beta = P(g(g(a)))$, namely $M_1^{\prec_B \beta} = \{P(a), P(g(a))\}$ and $M_2^{\prec_B \beta} = \{\neg P(a), \neg P(g(a))\}$. However, note that only $M_2^{\prec_B \beta}$ can be completed to an overall (infinite) model, for example to $\{\neg P(g^n(a)) \mid n \in \mathbb{Z}\}$.

Hence, $M_1^{\prec_B \beta}$ is not extendable. Thus, by Theorem 25, after finitely many applications of Grow, our HSCL run actually backtracks to a different model. Here, growing to $\beta' = P(g^3(a))$ is sufficient. Since we follow the strategy proposed in Theorem 27, we even learn the clause $\neg P(a)$, with $M_1^{\prec_B \beta} \not\models \neg P(a)$ and, thus, rule out any extension of M_1 as a model completely.

At the end of this run, by Theorem 19, HSCL has explored all partial models with respect to $\beta' = P(g^3(a))$. With respect to these limits, there is only one single partial model $M_1^{\prec_B \beta'} = \{\neg P(a), \neg P(g(a)), \neg P(g^2(a))\}$, which was present on the trail right before the use of Unstuck.

4 Discussion

We have shown that simultaneously searching for a refutation and enumerating all potential models, restricted to the current finite limit, can be effectively combined. All learned clauses either are non-redundant consequences or point to potential models. For the SCL family a refutation is obtained by a resolution [13, 8, 10] or paramodulation [16] proof. Dynamic completeness for saturation-based resolution or paramodulation calculi [1, 19] requires a notion of fairness. A run is fair, if any possible non-redundant inference by resolution or paramodulation is eventually performed. For SCL such a notion of fairness is not needed. A run with respect to some fixed, finite bound β either results in a refutation or a stuck state. Then Theorem 27 tells us that this stuck state can either be extended to an overall model, possibly in the infinite, or the respective partial model will be eventually ruled out by a conflict and learning a respective clause.

There are several directions for future research. First, investigating a procedure that takes a stuck state, or the overall content of N' out of some state $(\Gamma; N; U; \beta; N'; k; D)$ and checks for models for N. The model in a stuck state is a finite domain model, so all techniques developed

for finite domain models [11, 22, 15] can be explored to check whether the stuck state model is or can be extended to a finite model for the overall clause set. These techniques start with the size for a finite domain model and then try to define interpretations for predicates and functions in order to eventually satisfy the clause set. In our case we already have a finite domain model for a restricted number of ground instantiations but need to extend it to all potential ground instantiations. That means we have already done half the way of [11, 22, 15] and need to complete it. The techniques they developed, e.g., splitting clauses, are then immediately be applied for testing for the existence of such an extension. This will dramatically reduce the search space of these techniques [11, 22, 15] and still guarantee completeness. If this does not work we can check for an infinite domain extension of the finite stuck state model. This problem is undecidable, in general. However, for example, the following approach will work. Take a stuck state model candidate and extend it to an infinite domain model using an effective first-order model representation formalism, e.g., see [12]. Then check whether this results in a model for the overall clause set.

Second, any (ground) property that holds in all enumerated models is actually true, in general, and can therefore be added to the current set of clauses. Any property that holds at least in some enumerated models might be a good candidate to be tested for being true in all models. This test can be combined with the overall search for a refutation. The model exploration can also be explored towards satisfiability preserving inferences, by actually learning clauses that are true in some models and then searching for a refutation, similar to approaches in propositional [14] and first-order logic [6].

Acknowledgements: We thank our anonymous reviewers for their constructive comments.

References

[1] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier, 2001.

[2] Peter Baumgartner. Hyper tableau – the next generation. In Harrie C. M. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX '98, Oisterwijk, The Netherlands, May 5-8, 1998, Proceedings*, volume 1397 of *Lecture Notes in Computer Science*, pages 60–76. Springer, 1998.

[3] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Lemma learning in the model evolution calculus. In *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 572–586. Springer, 2006.

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009. Second Edition in 2021.

[5] Maria Paola Bonacina, Ulrich Furbach, and Viorica Sofronie-Stokkermans. On first-order model-based reasoning. In Narciso Martí-Oliet, Peter Csaba Ölveczky, and Carolyn L. Talcott, editors, *Logic, Rewriting, and Concurrency - Essays dedicated to José Meseguer on the Occasion of His 65th Birthday*, volume 9200 of *Lecture Notes in Computer Science*, pages 181–204. Springer, 2015.

[6] Maria Paola Bonacina, Christopher Lynch, and Leonardo Mendonça de Moura. On deciding satisfiability by dpll(g+) and unsound theorem proving. In Renate A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *LNCS*, pages 35–50. Springer, 2009.

[7] Maria Paola Bonacina and David A. Plaisted. Semantically-guided goal-sensitive reasoning: Model representation. *Journal of Automated Reasoning*, 56(2):113–141, 2016.

[8] Martin Bromberger, Alberto Fiori, and Christoph Weidenbach. Deciding the Bernays-Schoenfinkel fragment over bounded difference constraints by simple clause learning over theories. In Fritz Henglein, Sharon Shoham, and Yakir Vizel, editors, *Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17-19, 2021, Proceedings*, volume 12597 of *Lecture Notes in Computer Science*, pages 511–533. Springer, 2021.

[9] Martin Bromberger, Simon Schwarz, and Christoph Weidenbach. Exploring partial models with SCL. In Boris Konev, Claudia Schon, and Alexander Steen, editors, *Proceedings of the Workshop on Practical Aspects of Automated Reasoning Co-located with the 11th International Joint Conference on Automated Reasoning (FLoC/IJCAR 2022), Haifa, Israel, August, 11 - 12, 2022*, volume 3201 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

[10] Martin Bromberger, Simon Schwarz, and Christoph Weidenbach. SCL(FOL) revisited. *CoRR*, abs/2302.05954, 2023.

[11] Koen Claessen and Niklas Soerensson. New techniques that improve mace-style finite model finding. In *Proceedings of the CADE-19 Workshop: Model Computation - Principles, Algorithms, Applications*, 2003.

[12] Christian G. Fermüller and Reinhard Pichler. Model representation over finite and infinite signatures. *J. Log. Comput.*, 17(3):453–477, 2007.

[13] Alberto Fiori and Christoph Weidenbach. SCL clause learning from simple models. In *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, pages 233–249, 2019.

[14] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Short proofs without new variables. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2017.

[15] Mikolas Janota and Martin Suda. Towards smarter mace-style model finders. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, volume 57 of *EPiC Series in Computing*, pages 454–470. EasyChair, 2018.

[16] Hendrik Leidinger and Christoph Weidenbach. SCL(EQ): SCL for first-order logic with equality. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*, volume 13385 of *Lecture Notes in Computer Science*, pages 228–247. Springer, 2022.

[17] William McCune. A davis-putnam program and its application to finite first-order model search: Quasigroup existence problems. Technical report, Argonne National Laboratory, 1994.

[18] William McCune. Mace4 reference manual and guide. *CoRR*, cs.SC/0310055, 2003.

[19] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier, 2001.

[20] Juan Antonio Navarro Pérez and Andrei Voronkov. Proof systems for effectively propositional logic. In *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, pages 426–440. Springer, 2008.

[21] Ruzica Piskac, Leonardo Mendonça de Moura, and Nikolaj Bjørner. Deciding effectively propositional logic using DPLL and substitution sets. *Journal of Automated Reasoning*, 44(4):401–424, 2010.

[22] Giles Reger, Martin Suda, and Andrei Voronkov. Finding finite models in multi-sorted first-order logic. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 323–341. Springer, 2016.

[23] John Alan Robinson and Andrei Voronkov, editors. *Handbook of Automated Reasoning (in 2*

volumes). Elsevier and MIT Press, 2001.

[24] Olga Shumsky, Ralph W. Wilkerson, William McCune, and Fikret Erçal. Direct finite first-order model generation with negative constraint propagation heuristic. In Barrett R. Bryant, Janice H. Carroll, Dave Oppenheim, Jim Hightower, and K. M. George, editors, *Proceedings of the 1997 ACM symposium on Applied Computing, SAC'97, San Jose, CA, USA, February 28 - March 1*, pages 25–29. ACM, 1997.

[25] John K. Slaney. FINDER: finite domain enumerator - system description. In Alan Bundy, editor, *Automated Deduction - CADE-12, 12th International Conference on Automated Deduction, Nancy, France, June 26 - July 1, 1994, Proceedings*, volume 814 of *Lecture Notes in Computer Science*, pages 798–801. Springer, 1994.

5 Appendix

Theorem 7: Soundness of the HSCL rules

All HSCL rules preserve sound states.

Proof. As the hypothesis, assume a state $(\Gamma; N; U; \beta; N'; k; D)$ is sound. We show that any application of a rule results again in a sound state. For the conflict, resolve and backtrack rules we only show the extended versions, the original versions are similar.

$\Rightarrow^{\text{Decide}}_{\text{HSCL}}$. Assume Decide is applicable to the HSCL state $(\Gamma; N; U; \beta; N'; k; D)$, yielding a resulting state $(\Gamma, \text{D}{:}L\sigma^{k+1}; N; U; \beta; N'; k+1; D)$. Then there is an $L \in C$ for $C \in N \cup U$, $L\sigma$ is ground and undefined in Γ, and $L\sigma \prec_B \beta$. Also, there can be no active conflict, i.e. $D = \top$.

1, 4 By the precondition, $L\sigma$ is undefined in Γ (4). Hence, adding $\text{D}{:}L\sigma$ does not make Γ inconsistent (1).

2, 3, 5 Trivially fulfilled by hypothesis.

6, 7 Since $D = \top$, the rules are trivially satisfied.

8 For all literals $L'\sigma' \in \Gamma$, this holds by hypothesis. For $L\sigma$ this follows directly from the preconditions of the rule.

$\Rightarrow^{\text{Propagate}}_{\text{HSCL}}$. Assume Propagate is applicable to $(\Gamma; N; U; \beta; N'; k; D)$, yielding a resulting state $(\Gamma, \text{P}{:}L\sigma^{(C_0 \vee L)\delta\cdot\sigma}; N; U; \beta; N'; k; D)$. Then, there is a $C \vee L \in (N \cup U)$ such that $C = C_0 \vee C_1$, $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $(C \vee L)\sigma$ is ground, $(C \vee L)\sigma \prec_B \{\beta\}$, $C_0\sigma$ is false under Γ, and $L\sigma$ is undefined in Γ Also, there can be no active conflict, i.e. $D = \top$.

1, 4 By the precondition, $L\sigma$ is undefined in Γ (4). Hence, adding $\text{P}{:}L\sigma$ does not make Γ inconsistent (1).

2 Consider any decomposition $\Gamma, \text{P}{:}L\sigma^{(C_0 \vee L)\delta\cdot\sigma} = \Gamma_1, \text{P}{:}L'\sigma'^{C'_0 \vee L'\cdot\sigma'}, \Gamma_2$. In the case of $L'\sigma \neq L\sigma$, we can apply the hypothesis for the state $(\Gamma; N; U; \beta; N'; k; D)$. Hence, only the case $\Gamma_1 = \Gamma$, $L'\sigma' = L\sigma$, and $C'_0\sigma = C_0\sigma$ is left to prove.

First, note that $C_0\sigma$ is false under $\Gamma_1 = \Gamma$ by the preconditions. Also, $L\sigma$ must be undefined in Γ by the preconditions. Lastly, it needs to be shown that $N \cup U \models (C_0 \vee L)\delta$. Clearly, since $C \vee L \in (N \cup U)$, it holds that $N \cup U \models C \vee L$. Since $C = C_0 \vee C_1$ and $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$ it follows from the soundness of Factorization that $C \models (C_0 \vee L)$ and by this $N \cup U \models C_0 \vee L$.

3, 5 Follows trivially from the induction hypothesis.

6, 7 Since $D = \top$, the rules are trivially satisfied.

8 For all literals $L'\sigma' \in \Gamma$, this holds by hypothesis. For $L\sigma$, consider the precondition that $(C \vee L)\sigma \prec_B \{\beta\}$. By the definition of the multiset extension of $\prec_B$, it follows that $L\sigma \prec_B \beta$ must hold as well.

$\Rightarrow^{\text{Exclude}}_{\text{HSCL}}$. Assume Exclude is applicable to $(\Gamma; N; U; \beta; N'; k; D)$, yielding a resulting state $(\Gamma, \text{E}{:}L\sigma^{k+1:(C_0 \vee L)\delta\cdot\sigma}; N; U; \beta; N'; k+1; \top)$.

Then, there is a $C \vee L \in N'$, $C = C_0 \vee C_1$, $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $C_0\sigma$ is false under Γ, $(C \vee L)\sigma$ is ground, $(C \vee L)\sigma \prec_B \{\beta\}$, and $L\sigma$ is undefined in Γ. Also, there can be no active conflict, i.e. $D = \top$.

1, 4 By the precondition, $L\sigma$ is undefined in Γ (4). Hence, adding $\text{E}{:}L\sigma$ does not make Γ inconsistent (1).

2, 5 Follows trivially from the induction hypothesis.

3 Consider any decomposition

$$\Gamma, \text{E}{:}L\sigma^{k+1:(C_0 \vee L)\delta\cdot\sigma} = \Gamma_1, \text{E}{:}L'\sigma'^{k':C_0' \vee L'\cdot\sigma'}, \Gamma_2$$

In the case of $L'\sigma \neq L\sigma$, we can apply the hypothesis for the state $(\Gamma; N; U; \beta; N'; k; D)$. Hence, only the case $\Gamma_1 = \Gamma$, $L'\sigma' = L\sigma$, and $C_0'\sigma = C_0\sigma$ is left to prove.

First, note that $C_0\sigma$ is false under $\Gamma_1 = \Gamma$ by the preconditions. Also, $L\sigma$ must be undefined in Γ by the preconditions. Lastly, it needs to be shown that $N \cup N' \models (C_0 \vee L)\sigma$. Clearly, since $C \vee L \in N'$, it holds that $N \cup N' \models C \vee L$. $(C_0 \vee C_1 \vee L)\sigma$ is an instance of $C \vee L$. By the preconditions of Propagate, $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$. Hence, $C \models (C_0 \vee L)\sigma$ and by this $N \cup N' \models (C_0 \vee L)\sigma$.

6, 7 Since $D = \top$, the rules are trivially satisfied.

8 For all literals $L'\sigma' \in \Gamma$, this holds by hypothesis. For $L\sigma$, consider the precondition that $(C \vee L)\sigma \prec_B \{\beta\}$. By the definition of the multiset extension of $\prec_B$, it follows that $L\sigma \prec_B \beta$ must hold as well.

$\Rightarrow^{\text{ConflictE}}_{\text{HSCL}}$. Assume ConflictE is applicable to $(\Gamma; N; U; \beta; N'; k; D)$, yielding a resulting state $(\Gamma; N; U; \beta; N'; k; (C \cdot \sigma)_{\text{E}})$. Then, there is a $C \in N'$ such that $C\sigma$ false in Γ.

1-4, 8 Trivially fulfilled by hypothesis, as the trail Γ is not modified.

5 Follows trivially from the induction hypothesis, as U is not modified.

6 Since $D = (C \cdot \sigma)_{\text{E}}$, this rules is trivially satisfied.

7 It holds that $D = (C \cdot \sigma)_{\text{E}}$. By the preconditions of ConflictE, $C\sigma$ must be false under Γ. Furthermore, since $C \in N'$ it holds that $N' \models C$. Hence, clearly it is also the case that $N \cup N' \models C$. Lastly, it remains to show that $\text{gnd}^{\prec_B \beta}(N \cup N') \models C\sigma$. By soundness (8), we know that for all literals $L\mu \in \Gamma$ it holds that $L\mu \prec_B \beta$. Since $C\sigma$ is false in Γ, it must hold that all literals in $C\sigma$ are also $\prec_B \beta$. Combined with $N \cup N' \models C$, this yields that $\text{gnd}^{\prec_B \beta}(N \cup N') \models C\sigma$.

$\Rightarrow^{\text{Skip}}_{\text{HSCL}}$. Assume Skip is applicable to $(\Gamma = \Gamma', X{:}L; N; U; \beta; N'; k; (D \cdot \sigma)_u)$, yielding a resulting state $(\Gamma'; N; U; \beta; N'; k - i; (D \cdot \sigma)_u)$. By the preconditions of skip, it must hold that $\text{comp}(L)$ does not occur in $D\sigma$, and if $X \in \{D, E\}$, i.e. L is a decision or exclusion literal, then $i = 1$, else $i = 0$

1-4, 8 Directly fulfilled by hypothesis, as all prefixes of Γ still fulfil all properties. In particular, this holds for the prefix Γ' of Γ.

5 Follows trivially from the induction hypothesis, as U is not modified.

6, 7 After the application of Skip, $(D \cdot \sigma)_u$ is the current conflict. Since D is not modified, $N \models D$ (resp. $N \cup N' \models D$) and $\text{gnd}^{\prec_B \beta}(N) \models D\sigma$ (resp. $\text{gnd}^{\prec_B \beta}(N \cup N') \models D\sigma$) still hold by hypothesis. It is left to show that $D\sigma$ is false under the resulting Γ' under the assumption that $D\sigma$ is false under Γ. However, since $\text{comp}(L) \notin D\sigma$, this is trivially fulfilled, as the removal of $\text{comp}(L)$ from the trail Γ cannot make $D\sigma$ undefined. Hence, $D\sigma$ must be false under Γ' as well.

$\Rightarrow^{\text{Factorize}}_{\text{HSCL}}$. Assume Factorize is applicable to $(\Gamma; N; U; \beta; N'; k; ((D \vee L \vee L') \cdot \sigma)_u)$, yielding a resulting state $(\Gamma; N; U; \beta; N'; k; ((D \vee L)\eta \cdot \sigma)_u)$. Then, $L\sigma = L'\sigma$ and $\eta = \text{mgu}(L, L')$.

1-4, 8 Trivially fulfilled by hypothesis, as the trail Γ is not modified.

5 Follows trivially from the induction hypothesis, as U is not modified.

6, 7 After the application of Factorize, $((D \vee L)\eta \cdot \sigma)_u$ is the current conflict. W.l.o.g. assume we are in the $((D \vee L)\eta \cdot \sigma)_R$ case, i.e. the factorized clause is a regular conflict. By the hypothesis $N \models (D \vee L \vee L')$. From the preconditions of Factorize, $L\sigma = L'\sigma$ and $\eta = \text{mgu}(L, L')$. Thus, $(D \vee L \vee L')\eta$ is an instance of $(D \vee L \vee L')$ and $N \models (D \vee L \vee L')\eta$. Since $L\eta = L'\eta$, $(D \vee L \vee L')\eta \models (D \vee L')\eta$. Thus, $N \models (D \vee L)\eta$. By the preconditions, $\text{gnd}^{\prec_B \beta}(N) \models \text{gnd}^{\prec_B \beta}((L \vee L \vee L')\sigma)$. Hence, $(D \vee L \vee L')\sigma \prec_B \{\beta\}$. Thus, $(D \vee L)\eta\sigma = (D \vee L)\sigma \prec_B \{\beta\}$. From this, it follows that $\text{gnd}^{\prec_B \beta}(N) \models \text{gnd}^{\prec_B \beta}((D \vee L)\sigma)$.

Furthermore, $(D \vee L)\eta\sigma$ is false under Γ, since $(D \vee L)\eta\sigma = (D \vee L)\sigma$ by the definition of an mgu, and $(D \vee L \vee L')\sigma$ is already false under Γ.

$\Rightarrow^{\text{ResolveE}}_{\text{HSCL}}$. Assume ResolveE can be applied to $(\Gamma', X{:}L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; N'; k; ((D \vee L') \cdot \sigma)_{\text{E}})$ yielding a resulting state $(\Gamma', X{:}L\delta^{(C \vee L) \cdot \delta}; N; U; \beta; N'; k; ((D \vee C)\eta \cdot \sigma\delta)_{\text{E}})$

By the preconditions of ResolveE, it holds that $X \in \{\text{E}, \text{P}\}$.

1-4, 8 Trivially fulfilled by hypothesis, as the trail Γ is not modified.

5 Follows trivially from the induction hypothesis, as U is not modified.

6 Since $D = (C \cdot \sigma)_{\text{E}}$, this rule is trivially satisfied.

7 After the application of ResolveE, $((D \vee C)\eta \cdot \sigma\delta)_R$ is the current conflict.

By the hypothesis, $(D \vee L')\sigma$ is false under Γ. In particular, $D\sigma$ is false under Γ. By soundness (2), we know that $C\delta$ must be false under Γ as well. Hence, $(D \vee L)\eta\sigma\delta$ is false under Γ.

By the hypothesis, $N \cup N' \models (D \vee L')$. Since $(D \vee L')\eta$ is an instance of $(D \vee L')$, it holds that $N \cup N' \models (D \vee L')\eta$. Furthermore, by soundness (3) we know that $N \cup N' \models (C \vee L)$ By instantiation with η, it holds that $N \cup N' \models (C \vee L)\eta$. By the soundness of resolution, this implies $N \cup N' \models (D \vee C)\eta$.

Lastly, since $(D \vee L')\sigma$ is false in Γ, all occurring literals in $\{(D \vee L')\sigma\} \prec_B \{\beta\}$. With similar argumentation, $\{(C \vee L)\delta\} \prec_B \{\beta\}$. Hence, in particular, $(D \vee C)\eta\sigma\delta \prec_B \{\beta\}$ and, thus, $\text{gnd}^{\prec_B\beta}(N \cup N') \models \text{gnd}^{\prec_B\beta}((D \vee C)\eta\sigma\delta)$.

$\Rightarrow^{\text{BacktrackE}}_{\text{HSCL}}$. Assume BacktrackE is applicable to the state $(\Gamma = \Gamma', \Gamma''; N; U; \beta; N'; k; ((D \vee L) \cdot \sigma)_{\text{E}})$, yielding a resulting HSCL state $(\Gamma'; N; U; \beta; N' \cup \{D \vee L\}; k'; \top)$.

1-4, 8 Directly fulfilled by hypothesis, as all prefixes of Γ still fulfil all properties. In particular, this holds for the prefix Γ' of Γ.

5 Follows trivially from the hypothesis, as neither N nor U are modified.

6, 7 Since after an application of BacktrackR the conflict is resolved, i.e. $D = \top$, the rules are trivially satisfied.

$\Rightarrow^{\text{Unstuck}}_{\text{HSCL}}$. Assume Unstuck is applicable to $(\Gamma; N; U; \beta; N'; k; \top)$, yielding a resulting state $(\epsilon; N; U; \beta; N' \cup \{C\}; 0; \top)$.

1-4, 8 For the empty trail $\Gamma = \epsilon$, all properties follow directly as in Lemma 6.

5 Follows trivially from the hypothesis, as neither N nor U are modified.

6, 7 Since $D = \top$ the rules are trivially satisfied.

□

Lemma 11: Regular Conflict Resolution in HSCL

Consider a HSCL conflict state $(\Gamma, X{:}L; N; U; \beta; N'; k; (D)_u)$. In a regular run, during conflict resolution, at least the rightmost literal L is resolved with.

Proof. To prove the above claim, we distinguish the two cases how a conflict can be detected in HSCL. For the resulting conflict state, only the six conflict resolution rules Skip, Factorize, ResolveR, ResolveE, BacktrackR and BacktrackE can be applicable. To prove the claim of a resolution happening, we show that only Factorize and Resolve can be applied in a regular run to the resulting conflict state. Note that Factorize does neither remove literals from the trail nor remove any literal from the conflict clause. Hence, Factorize does not enable the application of any other rule. This shows that a Resolve step must happen at least once before any further conflict resolution rules (i.e. Skip, BacktrackR, BacktrackE) are applied.

The conflict D was either detected by the ConflictR rule. Then, it is of shape $(D)_R$. Otherwise, the conflict was detected by the ConflictE rule and is of shape $(D)_{\text{E}}$.

(Case ConflictR) If the rule Decide produced the state $(\Gamma, \text{D}{:}L; N; U; \beta; N'; k; \top)$ in a reasonable run, ConflictR is not immediately applicable. In case BacktrackR or BacktrackE produced the state $(\Gamma, X{:}L; N; U; \beta; N'; k; \top)$, there is the sequence of rule applications

$$\begin{aligned} &(\Gamma'_0, (L)_u, \Gamma_1, X : \text{comp}(L''\sigma)^{k'}; N; U'; \beta; N'; k'; ((D \vee L'') \cdot \sigma)_R) \\ \Rightarrow_{\text{Backtrack}\{\text{R,E}\}} &(\Gamma, X{:}L; N; U' \cup (D \vee L''); \beta; N'; k; \top) \end{aligned}$$

Then, by the definition of BacktrackR and BacktrackE, the newly learned clause $(D \vee L'')$ cannot be false with respect to Γ'_0, L. Thus, ConflictR is not applicable to $(D \vee L'')$. Furthermore, if there is a conflict to any other clause from $N \cup U$, by regularity, ConflictR must have been applied earlier in the run. In summary, L must be either a propagated or excluded literal.

However, it is not possible for L to be an excluded literal. If in a state $(\Gamma, \mathrm{E}{:}L; N; U; \beta; N'; k; \top)$ the rule ConflictR is applicable, then by reasonability the last rule cannot be Exclude, as Exclude may not enable an immediate application of ConflictR. Overall, L can neither be a decision nor an exclusion literal.
Then, BacktrackR is not applicable to $(\Gamma, \mathrm{D}{:}L; N; U; \beta; N'; k; (D)_R)$, as it requires L to be a decision or exclusion. Furthermore, L must occur in the conflict clause D. Otherwise, ConflictR could have been applied earlier to $(\Gamma; N; U; \beta; N'; k; \top)$, contradicting regularity. Hence, Skip is not applicable to our state. Overall, only Factorize and ResolveR can possibly be applied to our state. Factorize does neither modify the trail nor delete L from the conflict clause D. Thus, Factorize cannot enable any of the rules Skip or BacktrackR. Following from that, at least one application of ResolveR must take place in conflict resolution.

(Case ConflictE) This case works similar to the previous case. However, BacktrackE cannot be applied from a state $(\Gamma, \mathrm{E}{:}L; N; U; \beta; N'; k; \top)$. Hence, an application of ConflictE may directly follow an application of the Exclude rule. □

Theorem 13: Non-redundant learning in HSCL

Let $(\Gamma; N; U; \beta; N'; k; (C_0 \cdot \sigma_0)_u)$ be the state after an application of ConflictR (resp. ConflictE) in a regular run and let C be the clause learned at the end of the conflict resolution, then C is not redundant with respect to $N \cup U$ (resp. $N \cup N' \cup U$) and $\prec_\Gamma$.

Proof. Consider the following fragment of a derivation learning a clause implied by N:

$$\begin{array}{lll} \Rightarrow^{\text{ConflictR}}_{\text{HSCL}} & (\Gamma; N; U; \beta; N'; k; (C_0 \cdot \sigma_0)_R) & \\ \Rightarrow^{\{\text{Skip, Fact., Res.}\}^*}_{\text{HSCL}} & (\Gamma'; N; U; \beta; N'; k; (C \cdot \sigma)_R) & \Rightarrow^{\text{BacktrackR}}_{\text{HSCL}} \end{array}$$

By soundness $N \cup U \models C$ and $C\sigma$ is false under both Γ and Γ'. We prove that $C\sigma$ is non-redundant to $N \cup U$ with respect to $\prec_\Gamma$.

Assume there is an $S \subseteq \mathrm{gnd}(N \cup U)^{\preceq_\Gamma C\sigma}$ s.t. $S \models C\sigma$. There must be a clause $D \in S$ false under Γ, since all clauses in S have a defined truth value (as all undefined literals are greater in $\prec_\Gamma$ than all defined literals) and if $\Gamma \models S$ then $\Gamma \models C\sigma$ by transitivity of entailment, a contradiction.

By regularity, Γ must be of the shape $\Gamma = \Gamma'', L\delta^{C \vee L \cdot \delta}$, since no application of Decide can lead to an application of the rule ConflictR. Thus, the last applied rule must have been PropagateR. Furthermore, by Lemma 11, Resolve must have resolved at least the rightmost literal $L\delta$ from Γ. Thus, $L\delta \notin C\sigma$ and $\mathrm{comp}(L\delta) \notin C\sigma$. Since $D \prec_\Gamma C\sigma$, neither $L\delta$ nor $\mathrm{comp}(L\delta)$ may occur in D. However, this is a contradiction, since D is then already false under Γ'' and, thus, must have been chosen as a Conflict instance earlier in a regular run. Overall, there can be no $S \subseteq \mathrm{gnd}(N \cup U)^{\preceq_\Gamma C\sigma}$ with $S \models C\sigma$. Hence, $C\sigma$ is non-redundant to $N \cup U$ with respect to $\prec_\Gamma$.

Similarly, this result can be proven for learned clauses to N'. In contrast, a derivation learning a clause to N' with BacktrackE learns only non-redundant clauses with respect to $N \cup U \cup N'$ and $\prec_\Gamma$. □

Lemma 15: Correct Termination without Unstuck

If in a regular run no rule except Unstuck is applicable to an HSCL state $(\Gamma; N; U; \beta; N'; k; D)$, then either $D = (\bot)_R$, or $D = (\bot)_\mathrm{E}$, or $D = \top$ and $\Gamma \models \mathrm{gnd}^{\prec_B \beta}(N)$.

Proof. Consider a state $(\Gamma; N; U; \beta; N'; k; D)$ where $D \notin \{(\bot)_R, (\bot)_\mathrm{E}\}$.

Then, D can have one of the following shapes:

(Case $D = (C \cdot \sigma)_R$) then one of the rules ResolveR, Skip, Factorize or BacktrackR is applicable. First, consider the case of $\Gamma = \varepsilon$. By soundness, $C\sigma$ must be false under Γ. However, the only false clause under ε is $\bot$, a contradiction to $D \notin \{(\bot)_R, (\bot)_\mathrm{E}\}$. Thus, there is at least one literal on the trail. We split $\Gamma = \Gamma', X{:}L$ and distinguish on the source X of L:
If $X = P$, i.e. the top level literal is a propagated literal, then either ResolveR or Skip are applicable. In the case that $\mathrm{comp}(L)$ occurs in $C\sigma$, ResolveR is applicable. If $\mathrm{comp}(L) \notin C\sigma$, Skip is applicable.
For $X \in \{E, D\}$, i.e. the top level literal is a decision or exclusion literal, one of the rules Skip, BacktrackR, or Factorize is applicable. If $\mathrm{comp}(L)$ does not occur in $C\sigma$, then Skip can be applied. BacktrackR can be applied in all other cases if $C = (C' \vee \mathrm{comp}(L))$, where C' is of level $i' < k$. Note that for BacktrackR there must be a level j that is backtracked to. This level j always exists if all other preconditions are met. Hence, if Skip is not applicable, C is of the shape $C' \vee \mathrm{comp}(L)$. If C' is of level k, then Factorize can be applied instead, as C' must contain another instance of $\mathrm{comp}(L)$. Otherwise, C' is of level $i' < k$ and BacktrackR can be applied.

(Case $D = (C \cdot \sigma)_E$), then one of ResolveE, Skip, Factorize, or BacktrackE is applicable. This follows similarly to the previous case.

(Case $D = \top$) i.e. there is no conflict. Assume there are no undefined ground literals $L \prec_B \beta$ for $L \in C$, $C \in N \cup U$ in Γ. Now, either $\Gamma \models \mathrm{gnd}^{\prec_B \beta}(N)$ and thus Γ is already a partial model for N w.r.t. $\prec_B$ and β. Otherwise, if $\Gamma \not\models \mathrm{gnd}^{\prec_B \beta}(N)$ but all literals are defined, there must be a false clause $C \in \mathrm{gnd}^{\prec_B \beta}(N)$ which can be chosen as a ConflictR instance.
If there is at least one undefined ground literal $L \prec_B \beta$ occurring in $N \cup U$, one of the trail building rules Propagate, Decide, Exclude, ConflictR or ConflictE are applicable. Decide on the undefined ground literal L is always possible, as we only consider literals $L \in C$ for a $C \in (N \cup U)$. The application of Decide can, however, be restricted by reasonability.
If Decide on L is not applicable by reasonability, then $\Gamma, \mathrm{D}{:}L$ must lead to a direct application of ConflictR or ConflictE. Thus, there is a clause $D \in N \cup U \cup N'$ such that $D\sigma$ is false under $\Gamma, \mathrm{D}{:}L$. If $D\sigma$ is already false under Γ, then either ConflictR or ConflictE are applicable, depending on if $D \in N \cup U$ or $D \in N'$. Otherwise, D has the shape $D_0 \vee D_1$ where D_0 is false under Γ, and $D_1\sigma = \mathrm{comp}(L) \vee \cdots \vee \mathrm{comp}(L)$. Since D_0 is false under Γ, also $D_0 \prec_B \{\beta\}$ and since $L \prec_B \beta$ it holds that $D_0 \vee D_1 \prec_B \{\beta\}$. Hence, depending on if $D \in N \cup U$ or $D \in N'$, either Propagate or Exclude can be applied.
By a similar argumentation, if Exclude cannot be applied by reasonability, one of ConflictR, ConflictE, or Propagate can be applied. □

Theorem 19: Exhaustive stuck state exploration

Consider a HSCL run that ends in the final state $(\Gamma; N; U; \beta; N'; k; (\bot)_\mathrm{E})$. For all partial models M of N under $\prec_B$ and β, a stuck state corresponding to M is eventually explored in such a run.

Proof. Assume there is a partial model M such that no stuck state corresponding to M was visited. For M, by definition $M \models \mathrm{gnd}^{\prec_B \beta}(N)$ and $M \not\models \bot$. Since our run ended with $D = (\bot)_\mathrm{E}$, by soundness of the calculus it follows that $\mathrm{gnd}^{\prec_B \beta}(N \cup N') \models \bot$ in the final state. However, it cannot be the case that $\mathrm{gnd}^{\prec_B \beta}(N) \models \bot$, since otherwise by transitivity $M \models \mathrm{gnd}^{\prec_B \beta}(N) \models \bot$. Since initially, $N' = \emptyset$, there must be a HSCL rule application

$$(\Gamma; N; U; \beta; N'; k; D) \Rightarrow_{\text{HSCL}} (\Gamma; N; U; \beta; N' \cup \{C\}; k; D)$$

such that $M \models \text{gnd}^{\prec_B \beta}(N \cup N')$, but $M \not\models \text{gnd}^{\prec_B \beta}(N \cup N' \cup \{C\})$. This clause C can be added by two rules to N', $\Rightarrow_{\text{HSCL}}^{\text{BacktrackE}}$ or $\Rightarrow_{\text{HSCL}}^{\text{Unstuck}}$:

(Case BacktrackE) If C was added by $\Rightarrow_{\text{HSCL}}^{\text{BacktrackE}}$, then by soundness already $\text{gnd}^{\prec_B \beta}(N \cup N') \models C\sigma$. Thus, if $M \models \text{gnd}^{\prec_B \beta}(N \cup N')$ then also $M \models \text{gnd}^{\prec_B \beta}(N \cup N' \cup \{C\})$, a contradiction. Hence, it cannot be the case that C was added by $\Rightarrow_{\text{HSCL}}^{\text{BacktrackE}}$.

(Case Unstuck) By the preconditions of the rule, $(\Gamma; N; U; \beta; N'; k; D)$ must be a stuck state. It remains to show that this state corresponds to M. By Lemma 15, since $D \neq (\bot)_u$, it holds that Γ forms a partial model for N under $\prec_B$ and β. Thus, all ground literals $L\sigma$ occurring in N with $L\sigma \prec_B \beta$ are defined in Γ. Hence, it is only left to prove that $\Gamma \models M$. By induction over the trail size, we show that for each literal $L\sigma \in \Gamma$ it holds that $L\sigma \in M$. For the base case of $\Gamma = \epsilon$, nothing is to do. In the induction step, consider a trail decomposition $\Gamma' = \Gamma'', X{:}L\sigma$, where Γ' is a prefix of Γ. Then, $X{:}L\sigma$ was added to the trail by one of the following rules:

(Case Unstuck: Decide) Then, $\Gamma' = \Gamma'', \text{D}{:}L\sigma$. By the definition of $C = \bigvee_{L_i \in \text{decision}(\Gamma)} \text{comp}(L_i)$, it holds that $C = \text{comp}(L\sigma) \vee C'$, since $L\sigma$ is a decision in Γ'. Furthermore, all literals in C are already ground and $\prec_B \beta$. Hence, $\text{gnd}^{\prec_B \beta}(\{C\}) = \{\text{comp}(L\sigma) \vee C'\}$. Now, by assumption, $M \not\models \text{gnd}^{\prec_B \beta}(\{C\})$ and, thus, $M \not\models \text{comp}(L\sigma)$. Since M is a partial model that defines all ground literals $\prec_B \beta$, it must hold that $L\sigma \in M$.

(Case Unstuck: Propagate) This implies that the literal is of the shape $\text{P}{:}L\sigma^{(C_0 \vee L)\delta \cdot \sigma}$. By the preconditions of Propagate, there must be a clause $C_0 \vee C_1 \vee L \in (N \cup U)$ where $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, and $C_0\sigma$ is false under Γ''. By our induction hypothesis, we know that all literals defined in Γ'' are consistent with M. Now, assume that $\text{comp}(L\sigma) \in M$. But now, $C_0\sigma$ is false under M, and $C_1\sigma$ is false as well. Hence, the overall clause $(C_0 \vee C_1 \vee L)\sigma \in (N \cup U)$ is falsified under M. If $(C_0 \vee C_1 \vee L)\sigma \in N$, this directly contradicts the assumption that M is a partial model for N. If the clause is a learned clause, i.e. $(C_0 \vee C_1 \vee L)\sigma \in U$, by soundness of the calculus ($N \models U$), this leads to the same contradiction.

(Case Unstuck: Exclude) If the literal has the form $\text{E}{:}L\sigma_k^{(C_0 \vee L)\delta \cdot \sigma}$, with the same argumentation as in Propagate, there must be a clause $C_0 \vee C_1 \vee L \in N'$ where $C_1\sigma = L\sigma \vee \cdots \vee L\sigma$, and $C_0\sigma$ is false under Γ''. If we assume that $\text{comp}(L\sigma) \in M$, this falsifies the clause $(C_0 \vee C_1 \vee L)\sigma$ under M. However, since this clause is a ground clause $\prec_B \beta$, this contradicts the assumption that $M \models \text{gnd}^{\prec_B \beta}(N')$. □

Theorem 25: Eliminating non-extendable stuck states

Consider a regular HSCL run in a stuck state $(\Gamma; N; U; \beta; N'; k; \top)$. Then, either Γ is extendable to a model for N, or after finitely many applications of Grow, there is no more stuck state $(\Gamma'; N; U; \beta; N'; k; \top)$ reachable in this HSCL run, where $\Gamma \subseteq \Gamma'$.

Proof. Assume that Γ is not extendable to a model for N. Then, the clause set $N' = N \cup \{D \mid D \in \Gamma\}$, where all literals from Γ are added as unit clauses, is unsatisfiable. By compactness, as N' is unsatisfiable, there is some ground finite $N'' \subseteq \text{gnd}(N)$ that is unsatisfiable. Select β' in a way that all literals from N'' are $\prec_B$-smaller than β. By definition of $\prec_B$, there are only finitely many literals $\beta_0, \ldots, \beta_k \prec_B \beta'$. Hence, after k applications of Grow, the newly chosen β'' is $\prec_B$-larger than β'.

A stuck state of HSCL is explored when no rules except Unstuck, Restart and Grow are applicable. By Lemma 15, this entails that for a trail Γ' in this stuck state, $\Gamma' \models \mathrm{gnd}^{\prec_B \beta''}(N)$. However, as $\beta' \prec_B \beta''$ and $N'' \subseteq \mathrm{gnd}^{\prec_B \beta'}(N)$, it holds that $N'' \subseteq \mathrm{gnd}^{\prec_B \beta''}(N)$. Thus, $\mathrm{gnd}^{\prec_B \beta''}(N)$ is already unsatisfiable and hence no such Γ' can exist.

□

Theorem 27: Learning from non-extendable models

Consider a regular HSCL run in a stuck state $(\Gamma; N; U; \beta; N'; k; \top)$, where Γ is not extendable to a model. Then, after finitely many applications of Grow, HSCL can either find a refutation or learn a clause C to U that prevents Γ from being explored again, i.e. $\Gamma \not\models C\sigma$.

HSCL is guaranteed to learn such a clause by following a regular, fair strategy that does neither apply Restart nor Unstuck after encountering the stuck state.

Proof. As our run is fair and regular, if N is unsatisfiable, by Theorem 26 HSCL will refute the clause set and terminate. Hence, assume N is satisfiable, and thus HSCL cannot end up in $(\Gamma; N; U; \beta; N'; k; (\bot)_R)$.

Consider HSCL in the stuck state $(\Gamma; N; U; \beta; N'; k; \top)$, where Γ is not extendable to a model. By our strategy, neither Restart nor Unstuck are applicable. By the definition of a stuck state, this means that only Grow is applicable.

In the following, we show that a HSCL run after the application of Grow eventually must backtrack with BacktrackR to a proper prefix of Γ. To this end, consider the following fragment of a derivation:

$$\begin{array}{ll} & (\Gamma; N; U; \beta; N'; k; \top) \\ \Rightarrow_{\mathrm{HSCL}}^{\mathrm{Grow}} & (\Gamma; N; U; \beta'; N'; k; \top) \\ \Rightarrow_{\mathrm{HSCL}}^{*} & (\Gamma'', K, \Gamma'; N; U'; \beta'; N'; k; (C \cdot \sigma)_R) \\ \Rightarrow_{\mathrm{HSCL}}^{\mathrm{BacktrackR}} & (\Gamma''; N; U' \cup \{C\}; \beta'; N'; k; \top) \end{array}$$

In this fragment, note that by the preconditions of BacktrackR, the learned clause $C\sigma$ must be false under Γ'', K. As Γ'' is a proper prefix of Γ, Γ'', K is a prefix of Γ as well. Hence, $\Gamma \not\models C\sigma$.

It is left to show that the continued derivation will always follow the above structure. To this end, note that ConflictE is not applicable to the stuck state by definition. Furthermore, note that ConflictE cannot be enabled by adding further literals to the trail directly after Grow, as such literals cannot yet occur in N' right after Grow, and can only be added via Unstuck, which is disabled. Thus, any state where $D = (C)_{\mathrm{E}}$ cannot occur.

Moreover, note that the HSCL run cannot end up with $D = \top$ and in a state where Γ is a prefix of the trail, as this would contradict Theorem 15. Furthermore, HSCL cannot spend infinitely many steps with Γ as a prefix of the trail by Theorem 25. Hence, after finitely many steps, HSCL must backtrack to a proper prefix of the trail. This can only happen during a conflict resolution of a regular conflict. As N is satisfiable, this conflict resolution must end with a BacktrackR application. □