3.16.4 SCL Clause Learning from Simple Models

The SCL calculus is similar to the NRCL calculus but more fine-grained. Instead of starting with all constants, it starts only with a finite subset of constants B that can be used for building models and driving reasoning. The SCL calculus can be extended to a calculus considering theories, see Section **??**.

The signature considered for the BS fragment is $\Sigma = (\mathcal{S}, \Omega, \Pi)$ where the finite set Ω only consists of constant symbols and for the set B considered by the calculus we require $B \subseteq \Omega$. A state is a five tuple $(M; N; U; B; k; C)$ where M is a trail of ground literals, N the set of input clauses, U the set of learned clauses, B a non-empty set of constants, k the current decision level and C a

clause that is $\top$ in case of searching for a model, $\bot$ in case a refutation has been found and different from $\top$, $\bot$ if C is a non-empty clause that is false in M.

Propagate $(M; N; U; B; k; \top) \Rightarrow_{\text{SCL}} (M, L\sigma^{(C_0 \vee L)\delta \cdot \sigma}; N; U; B; k; \top)$
provided $C \in (N \cup U)$, σ is grounding for C, $C = C_0 \vee C_1 \vee L$, $C_1\sigma = L\sigma \vee \ldots \vee L\sigma$, $C_0\sigma$ does not contain $L\sigma$, δ is the mgu of the literals in C_1 and L, $M \models \neg(C_0\sigma)$, $\text{codom}(\sigma) \subseteq B$, and $L\sigma$ is undefined in M

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.

Decide $(M; N; U; B; k; \top) \Rightarrow_{\text{SCL}} (M, L\sigma^{k+1}; N; U; B; k+1; \top)$
provided $L\sigma$ is undefined in M, $|L\sigma| \in \text{atoms}(\text{grd}((\mathcal{S}, B, \Pi), N \cup U))$, and $\text{codom}(\sigma) \subseteq B$

The number of potential trails of a run is finite because the rules Propagate and Decide make sure that only undefined literals over a fixed finite sequence B of constants are added to the trail.

Conflict $(M; N; U; B; k; \top) \Rightarrow_{\text{SCL}} (M; N; U; B; k; D \cdot \sigma)$
provided $D \in (N \cup U)$, σ is grounding for D, $M \models \neg(D\sigma)$, and $\text{codom}(\sigma) \subseteq B$

Resolve $(M, L\rho^{C \vee L \cdot \rho}; N; U; B; k; (D \vee L') \cdot \sigma) \Rightarrow_{\text{SCL}} (M, L\rho^{C \vee L \cdot \rho}; N; U; B; k; (D \vee C)\eta \cdot \sigma\rho)$
provided $L\rho = \text{comp}(L'\sigma)$, and $\eta = \text{mgu}(L, \text{comp}(L'))$

Note that Resolve does not remove the literal $L\rho$ from the trail. This is needed if the clause $D\sigma$ contains further literals complementary of $L\rho$ that have not been factorized.

Factorize $(M; N; U; B; k; (D \vee L \vee L') \cdot \sigma) \Rightarrow_{\text{SCL}} (M; N; U; B; k; (D \vee L)\eta \cdot \sigma)$
provided $L\sigma = L'\sigma$, and $\eta = \text{mgu}(L, L')$

Note that Factorize is not limited with respect to the trail. It may apply to any two literals that become identical by application of the grounding substitution σ.

Skip $(M, L; N; U; B; k; D \cdot \sigma) \Rightarrow_{\text{SCL}} (M; N; U; B; l; D \cdot \sigma)$
provided $\text{comp}(L)$ does not occur in $D\sigma$, if L is a decision literal then $l = k - 1$, otherwise $l = k$

Note that Skip can also skip decision literals. This is needed because I don't eventually require exhaustive propagation. While exhaustive propagation in

CDCL is limited to the number of propositional variables, in the context BS it is already in the size of the input clause set.

Backtrack $(M, K^{i+1}, M'; N; U; B; k; (D \vee L)\cdot\sigma) \Rightarrow_{\mathrm{SCL}} (M, L\sigma^{(D\vee L)\cdot\sigma}; N; U \cup \{D \vee L\}; B; i; \top)$
provided $L\sigma$ is of level k, and $D\sigma$ is of level i

The definition of Backtrack requires that $L\sigma$ is the only literal of level k in $(D \vee L)\sigma$. Additional occurrences of $L\sigma$ in D have to be factorized first before Backtrack can be applied.

Grow $(M; N; U; B; k; \top) \Rightarrow_{\mathrm{SCL}} (\epsilon; N; U; B \cup B'; 0; \top)$
provided $B' \subset \Omega$ is a non-empty sequence of constants of distinct from the constants in B

Example 3.16.5 (Comparing Proof Length Depending on Unit Clause Propagation)**.** Proofs generated without full propagation can be exponentially shorter than proofs generated by exhaustive propagation. Consider the simple BS clause set over constants $\Omega = \{a, b\}$

$$N = \{R(x_1, \ldots, x_n, a, b), P \vee Q, P \vee \neg Q, \neg P \vee Q, \neg P \vee \neg Q\}$$

A run without exhaustive propagation can ignore generating the 2^n different ground instances of $R(x_1, \ldots, x_n, a, b)$ starting with initial set $B = \{a, b\}$. Instead it refutes the propositional part of N in the usual CDCL style by starting with a decision on P or Q. For the example it is obvious that the instances of $R(x_1, \ldots, x_n, a, b)$ can be ignored, but in general it is not.

Consider another example, taken from [**?**], where exhaustive propagation leads to exponentially longer proofs compared to the shortest resolution proof.

Example 3.16.6 (Comparing Proof Length Depending on Clause Propagation)**.** Let i be a positive integer and consider the clause set N^i with one predicate P of arity i consisting of the following clauses, where we write $\bar{x}, \bar{0}$ and $\bar{1}$ to denote sequences of the appropriate length of variables and constants to meet the arity of P:

$$P(\bar{0}) \qquad \neg P(\bar{1})$$

and i clauses of the form $\neg P(\bar{x}, 0, \bar{1}) \vee P(\bar{x}, 1, \bar{0})$
where the length of $\bar{1}$ varies between 0 and $i - 1$. The example encodes an i-bit counter. An SCL run with exhaustive propagation on this clause set finds a conflict after $O(2^i)$ propagations without any application of Decide.

For the instance $i = 4$ we get the clauses of N^4:

$$N^4 = \{ \begin{array}{l} 1 : P(0,0,0,0) \\ 2 : \neg P(x_1, x_2, x_3, 0) \vee P(x_1, x_2, x_3, 1) \\ 3 : \neg P(x_1, x_2, 0, 1) \vee P(x_1, x_2, 1, 0) \\ 4 : \neg P(x_1, 0, 1, 1) \vee P(x_1, 1, 0, 0) \\ 5 : \neg P(0, 1, 1, 1) \vee P(1, 0, 0, 0) \\ 6 : \neg P(1, 1, 1, 1) \end{array} \}$$

For this clause set an SCL all unit clauses from $P(0,0,0,0)$ to $P(1,1,1,1)$ via 2^4 applications of Propagate, then finds a conflict with clause 6 and then uses 2^4 times Resolve to end up in $\bot$.

Instead a short resolution refutation can be obtained by

2.2 Res 3.1	$7 : \neg P(x_1, x_2, 0, 0) \vee P(x_1, x_2, 1, 0)$
7.2 Res 2.1	$8 : \neg P(x_1, x_2, 0, 0) \vee P(x_1, x_2, 1, 1)$
8.2 Res 4.1	$9 : \neg P(x_1, 0, 0, 0) \vee P(x_1, 1, 0, 0)$
9.2 Res 8.1	$10 : \neg P(x_1, 0, 0, 0) \vee P(x_1, 1, 1, 1)$
10.2 Res 5.1	$11 : \neg P(0, 0, 0, 0) \vee P(1, 0, 0, 0)$
11.2 Res 10.1	$12 : \neg P(0, 0, 0, 0) \vee P(1, 1, 1, 1)$
12.1 Res 6.1	$13 : \bot$

In general, $O(2i)$ many resolution steps are sufficient to refute N^i. This derivation can be simulated by SCL if exhaustive propagation is not used. For example, the first resolution step between clauses 2.2 and 3.1 can be simulated by first deciding $P(1,1,0,0)$ and $\neg P(1,1,1,0)$ yielding the state

$$([P(1,1,0,0)^1, \neg P(1,1,1,0)^2]; N; \emptyset; \{0,1\}; 2; \top)$$

now we can propagate using $\neg P(1,1,1,0)^2$ with clause 3

$$([P(1,1,0,0)^1, \neg P(1,1,1,0)^2, \neg P(1,1,0,1)^{\neg P(x_1,x_2,0,1) \vee P(x_1,x_2,1,0) \cdot \{x_1 \mapsto 1, x_2 \mapsto 1\}}]; N; \emptyset; \{0,1\}; 2; \top)$$

and then get a conflict with clause 2 by closure $(\neg P(x_1, x_2, x_3, 0) \vee P(x_1, x_2, x_3, 1)) \cdot \{x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 0\}$. Next we apply Conflict and Resolve to the rightmost propagated literal and get

$$([P(1,1,0,0)^1, \neg P(1,1,1,0)^2]; N; \emptyset; \{0,1\}; 2; (\neg P(x_1, x_2, 0, 0) \vee P(x_1, x_2, 1, 0)) \cdot \{x_1 \mapsto 1, x_2 \mapsto 1\}).$$

Finally Backtrack is applicable resulting in

$$([P(1,1,0,0)^1, P(1,1,1,0)^{(\neg P(x_1,x_2,0,0) \vee P(x_1,x_2,1,0)) \cdot \{x_1 \mapsto 1, x_2 \mapsto 1\}}]; N; \{\neg P(x_1, x_2, 0, 0) \vee P(x_1, x_2, 1, 0)\}; \{0,$$

However, to continue with the proof we need also need a Restart rule, which is anyway needed for completeness to get out of stuck states.

Definition 3.16.7 (Rule Types)**.** The rules Propagate, Decide, Grow, and Conflict are called *conflict search* rules and the rules Resolve, Skip, Factorize, and Backtrack are called *conflict resolution* rules.

Definition 3.16.8 (Well-formed States)**.** A state $(M; N; U; B; k; D)$ is *well-formed* if the following conditions hold:

1. all constants appearing in $(M; N; U; B; k; D)$ are from B or occur in N.
2. M is satisfiable

3. $N \models U$,
4. Propagating clauses remain propagating and conflict clauses remain false:
   (a) if $D = C \cdot \sigma$ then $C\sigma$ is false in M,
   (b) if $M = M_1, L\sigma^{(C \vee L) \cdot \sigma}, M_2$ then $C\sigma$ is false in M_1, and $L\sigma$ is undefined in M_1.

Lemma 3.16.9 (Rules preserve Well-Formed States)**.** The rules of SCL preserve well-formed states.

Definition 3.16.10 (Stuck State)**.** A state $(M; N; U; B; k; D)$ is called *stuck* if $D \neq \bot$ and none of the rules Propagate, Decide, Conflict, Resolve, Factorize, Skip, or Backtrack is applicable.

Proposition 3.16.11 (Form of Stuck States)**.** If a run (without rule Grow) where Conflict was applied eagerly ends in a stuck state $(M; N; U; B; k; D)$, then $D = \top$ and all ground literals that can be build from the literals in N by instantiation with constants from B are defined in M.

Proof. First we prove that stuck states never appear during conflict resolution. Consider a well-formed state $(M; N; U; B; k; D \cdot \delta)$, we prove by case analysis that either Skip, Resolve, Factorize or Backtrack can be applied. If $M = M', L\sigma$ and $L\sigma$ is a literal such that $\text{comp}(L\sigma)$ is not contained in $D\delta$ then Skip can be applied. If $M = M', L\sigma^{C \cdot \sigma}$ with $D\delta = D' \vee \text{comp}(L\sigma)$ then Resolve can be applied. If $M = M', L\sigma^k, M''$ and D' contains multiple occurrences of $\text{comp}(L\sigma)$ then Factorize can be applied. In summary, we can reach a state with a unique literal $L\delta$ of level k in $D\delta$. Then Backtrack is applicable. Finally, if in some state $(M; N; U; B; k; \top)$ where Conflict is not applicable, some atom $|L| \in \text{atoms}(\text{grd}((\mathcal{S}, B, \Pi), (N \cup U)))$ is undefined, we can always apply Decide. □

Lemma 3.16.12 (Stuck States Produce Ground Models)**.** Every stuck state $(M; N; U; B; k; \top)$ produces a ground model, i.e., $M \models \text{grd}((\mathcal{S}, B, \Pi), (N \cup U)))$.

Proof. By contradiction. Consider any clause $C\sigma \in \text{grd}((\mathcal{S}, B, \Pi), (N \cup U))$. It can only be not true if $M \models \neg(C\sigma)$, because all literals are defined Proposition 3.16.11. But then Conflict would be applicable, a contradiction. □

Lemma 3.16.13 (Soundness)**.** If a derivation reaches the state $(M; N; U; B; k; \bot)$, then N is unsatisfiable.

Definition 3.16.14 (Reasonable Run)**.** A sequence of SCL rule applications is called a *reasonable run* if an application of rule Decide does not enable an application of rule Conflict.

Proposition 3.16.15 (Avoiding Conflicts after Decide)**.** Let N be a set of clauses and $(M; N; U; B; k; \top)$ be a state derived from $(\epsilon; N; \emptyset; B; 0; \top)$. If an application of rule Decide to $(M; N; U; B; k; \top)$ enables an application of rule Conflict, then Propagate would have been applicable to $(M; N; U; B; k; \top)$.

Definition 3.16.16 (Regular Run)**.** A sequence of SCL rule applications is called a *regular run* if it is a reasonable run, the rule Conflict has precedence over all other rules, and Resolve resolves away at least the rightmost literal from the trail.

Proposition 3.16.17 (Stuck States at Regular Runs)**.** Lemma 3.16.12 also holds for regular runs.

Corollary 3.16.18 (Regular Conflict Resolution)**.** Let N be a set of (BS) clauses. Then any conflict in an SCL regular run admits a regular conflict resolution if the run starts from state $(\epsilon; N; \emptyset; B; 0; \top)$.

Proposition 3.16.19 (Decide Creates no Conflict in Regular Runs)**.** Let N be a set of clauses. Then any application of Decide in an SCL regular run from starting state $(\epsilon; N; \emptyset; B; 0; \top)$ does not create a conflict.

Proof. Assume the contrary: then Propagate would have been applicable before Decide, contradicting with the definition of a regular and hence reasonable run. □

Corollary 3.16.20 (Conflicts Admit Regular Conflict Resolution)**.** Let N be a set of clauses. Then any conflict in an SCL regular run from starting state $(\epsilon; N; \emptyset; B; 0; \top)$ admits a regular conflict resolution.

Proof. We need to prove that it is possible to apply Resolve during conflict resolution. By Proposition 3.16.19 the rightmost foreground literal on the trail is a propagation literal and by regularity we know that this literal appears in the conflict clause. So a conflict resolution can start by skipping over the background literals and then resolving once with the rightmost foreground literal. □

Lemma 3.16.21 (Non-Redundant Clause Learning)**.** Let N be a set of clauses, and let $D \vee L$ be a clause learned in an SCL regular run such that $(\epsilon; N; \emptyset; B; 0; \top) \Rightarrow^*_{\text{SCL}} \Rightarrow^{\text{Backtrack}}_{\text{SCL}} (M, L\sigma^{(D \vee L)\cdot\sigma}; N; U \cup \{D \vee L\}; B; i; \top)$. Then $D \vee L$ is not redundant with respect to any ordering $\prec$ induced by the trail M.

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

$$
\begin{array}{ll}
\Rightarrow^{\text{Conflict}}_{\text{SCL}} & (M''; N; U; B; k; C_0 \cdot \sigma_0) \\
\Rightarrow^{\{\text{Skip, Factorize, Resolve}\}^*}_{\text{SCL}} & (M, K^{i+1}, M'; N; U; B; k; C_n \cdot \sigma_n) \\
\Rightarrow^{\text{Backtrack}}_{\text{SCL}} & (M, L\sigma^{(D \vee L)\cdot\sigma}; N; U \cup \{D \vee L\}; B; i; \top).
\end{array}
$$

where $C_n = D \vee L$ and $\sigma = \sigma_n$. Let $\prec$ be any induced by M. We prove that $C_n\sigma$ is not redundant with respect to $\prec$, B, and $(N \cup U)$. By soundness of resolution $(N \cup U) \models C_n$ and $C_n\sigma$ is false under both M and M, K^{i+1}, M', Lemma 3.16.8. For a proof by contradiction, assume there is a $N' \subseteq \text{grd}((\mathcal{S}, B, \Pi), (N \cup U)^{\preceq C_n\sigma}$ such that $N' \models C_n\sigma$. As $C_n\sigma$ is false under M, there is a ground clause $C' \in N'$ with $C' \preceq C_n\sigma$, and all literals from C' are defined in M and false by the definition of $\prec$.

The clause $C_0\sigma_0$ has at least one literal of level k and due to a regular run, Definition 3.16.16, the rightmost trail literal is resolved away in $C_n\sigma$, Corollary 3.16.20. Therefore, the rightmost foreground literal does not appear in C', so by regularity C' would have created a conflict at a previous state. □

Of course, in a regular run the ordering of literals on the trail will change, i.e., the ordering underlying Lemma 3.16.21 will change as well. Thus the non-redundancy property of Lemma 3.16.21 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. However, the ordering underlying Lemma 3.16.21 also entails a fixed subset ordering that is invariant against changes on the overall ordering. This means that our dynamic ordering entails non-redundancy criteria based on subset relations including forward redundancy. From an implementation perspective, this means that learned clauses need not to be tested for forward redundancy. Current resolution, or superposition based provers spent a reasonable portion of their time in testing forward redundancy of newly generated clauses. In addition, also tests for backward reduction can be restricted knowing that learned clauses are not redundant.

Lemma 3.16.22 (Termination of SCL)**.** Let N be a set of clauses and B be a finite set of background constants. Then any regular run with start state $(\epsilon; N; \emptyset; B; 0; \top)$ that uses Grow only finitely often terminates.

Proof. Since Grow can only be used a finite number of times we consider as a start state the state after the final application of Grow and prove termination of runs that never use Grow. We do so by giving an explicit termination measure on the SCL states. Given a state $(M; N; U; B; k; D)$ we define a termination measure μ as $\mu(M; N; U; B; k; D) = (u, s, m, r, d) \in \mathbb{N}^5$ with a lexicographical combination of $>$ where

- $l = |\operatorname{atoms}(\operatorname{grd}((\mathcal{S}, B, \Pi), (N \cup U)))|$, $u = 3^l - |\operatorname{grd}((\mathcal{S}, B, \Pi), U)|$, and $m = |M|$,
- in the case $D = \top$:
  - $s = 1 + l - m$, $d = 0$, and $r = 0$,
- otherwise if $D = D' \cdot \delta$:
  - $s = 0$,
  - if $M = M', L$ with then r is the number of copies of L in $D'\delta$
  - d is the number of literals in D'

The number of ground atoms $l = |\operatorname{atoms}(\operatorname{grd}((\mathcal{S}, B, \Pi), (N \cup U)))|$ is an upper bound to the length of the trail because the trail is consistent and no literal can appear more than once on the trail. Similarly, every learned clause has at least one non-redundant ground instance so $|\operatorname{grd}((\mathcal{S}, B, \Pi), U)|$ increases whenever SCL(T) learns a new clause and 3^l is an upper bound to the ground instances of

all learned clauses in a regular run. This means that Backtrack strictly decreases u, Decide, Propagate, and Conflict strictly decrease s without modifying u, Skip strictly decreases m without modifying u or s, Resolve strictly decreases r without modifying u, s, or m, and finally Factorize strictly decreases d, possibly decreases r and does not modify u, s, or m. □

Finally, we show that an unsatisfiable clause set can be refuted by SCL with any regular run if we start with a sufficiently large sequence of constants B and apply Decide in a fair way. In addition, we need a Restart rule to recover from a stuck state.

Restart $\quad (M; N; U; B; k; \top) \Rightarrow_{\mathrm{SCL}} (\epsilon; N; U; B; 0; \top)$

Of course, an unrestricted use of rule Restart immediately leads to non-termination.

Theorem 3.16.23 (Refutational Completeness of SCL)**.** Let N be an unsatisfiable clause set. Then any regular SCL run will derive the empty clause provided (i) Rule Grow and Decide are operated in a fair way, such that all possible trail prefixes of all considered sets B during the run are eventually explored, and (ii) Restart is only applied to stuck states.

Proof. If N is unsatisfiable then by compactness of fist-order logic, there exists a a finite set $N' = \{\Lambda_1 \parallel C_1, \ldots, \Lambda_n \parallel C_n\}$ of variable renamed copies of clauses from N and a finite set $B \subseteq \Omega$ of constants and a substitution σ, grounding for N' where $\mathrm{codom}(\sigma) = B$ such that $\bigwedge_i C_i\sigma$ is unsatisfiable. If the SCL rules are applied in a fair way, then they will in particular produce trails solely consisting of literals from $N'\sigma$. For these trails the states corresponding to these trails cannot end in a stuck state, because this contradicts the unsatisfiability of $\bigwedge_i C_i\sigma$. Instead, they all end in a conflict with some clause in $N'\sigma$. In addition, there are only finitely many such trails, because the number of literals in $N'\sigma$ is finite. Now let $\mu((M; N; U; B; k; \top))$ be the multiset of the levels of all states with trails from $N'\sigma$ until a conflict occurs. Each time a state with a trail from $N'\sigma$ results in a conflict, SCL learns a non-redundant clause that propagates at a strictly smaller level, Lemma 3.16.21. Thus $\mu((M; N; U; B; k; \top))$ strictly decreases after each Backtrack step after a conflict on a trail with atoms from $N'\sigma$. The clause learnt at level zero is the empty clause. □

Condition (i) of the above theorem is quite abstract. It can, e.g., be made effective by applying rule Grow only after all possible trail prefixes with respect to the current set B have been explored and to make sure that Decide does not produce the same stuck state twice.