SCL Clause Learning from Simple Models

The SCL calculus is simillar 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 8.14.

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 $(M; N; U; B; k; \top) \Rightarrow_{\mathsf{SCL}}$
$(M, L\sigma^{(C_0 \vee L)\delta \cdot \sigma}; N; U; B; k; \top)$

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, $M \models \neg(C_0\sigma)$, $\mathrm{codom}(\sigma) \subseteq B$, 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 $(M; N; U; B; k; \top) \Rightarrow_{\mathsf{SCL}}$
$(M, L\sigma^{k+1}; N; U; B; k+1; \top)$

provided $L\sigma$ is undefined in M,
$|L\sigma| \in \mathsf{atoms}(\mathsf{grd}((\mathcal{S}, B, \Pi), N \cup U))$, and $\mathsf{codom}(\sigma) \subseteq B$

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

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

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

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 $(M, L; N; U; B; k; D \cdot \sigma) \Rightarrow_{\mathsf{SCL}} (M; N; U; B; l; D \cdot \sigma)$
provided $\mathrm{comp}(L)$ 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 $(M, K^{i+1}, M'; N; U; B; k; (D \vee L) \cdot \sigma) \Rightarrow_{\mathsf{SCL}} (M, L\sigma^{(D \vee L) \cdot \sigma}; N; U \cup \{D \vee L\}; B; i; \top)$

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 $(M; N; U; B; k; \top) \Rightarrow_{\mathsf{SCL}} (\epsilon; N; U; B \cup B'; 0; \top)$

provided $B' \subset \Omega$ is a non-empty sequence of constants of distinct from the constants in B

Examples: Exponential Proof Length

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

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

3.16.7 Definition (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.

3.16.8 Definition (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:
   - 1..1 if $D = C \cdot \sigma$ then $C\sigma$ is false in M,
   - 2..2 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.

3.16.9 Lemma (Rules preserve Well-Formed States)

The rules of SCL preserve well-formed states.

3.16.10 Definition (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.

3.16.11 Proposition (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.

3.16.12 Lemma (Stuck States Produce Ground Models)

Every stuck state $(M; N; U; B; k; \top)$ produces a ground model, i.e., $M \models \mathrm{grd}((\mathcal{S}, B, \Pi), (N \cup U)))$.

3.16.13 Lemma (Soundness)

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

3.16.14 Definition (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.

3.16.15 Proposition (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 $(M; N; U; B; k; \top)$.

3.16.16 Definition (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.

3.16.17 Proposition (Stuck States at Regular Runs)

Lemma 3.16.12 also holds for regular runs.

3.16.18 Corollary (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 $(\epsilon; N; \emptyset; B; 0; \top)$.

3.16.19 Proposition (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.

3.16.20 Corollary (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.

3.16.21 Lemma (Non-Redundant Clause Learning)

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

3.16.22 Lemma (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.

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

3.16.23 Theorem (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.