SCL: Clause Learning from Simple Models

Lifting CDCL to first-order logic

Hi
CDCL – quo vadis?

\[ N = \{ P \lor Q, \; P \lor \neg Q, \; \neg P \lor Q, \; \neg P \lor \neg Q \} \]

State: \((\Gamma, N, U, k, 0)\)

\((\varepsilon, N, \emptyset, 0, T)\)

\[ \Rightarrow \]

Decide \((P^1, N, \emptyset, 1, T)\)

\[ \Rightarrow \]

Propagate \((P \lor Q, N, \emptyset, 1, T)\)

\[ \Rightarrow \]

Consolidate \((P^1, \neg P \lor \neg Q)\)

\[ \Rightarrow \]

Resolve \((P^1, N, \emptyset, 1, \neg P \lor \neg Q)\)

\[ \Rightarrow \]

Tandem \((\neg P \lor P, N, \emptyset, 1, T)\)
SCL Clause Learning from Simple Models

The basic idea of SCL is to lift the principles of CDCL, Section 2.9, to first-order logic:

1. operating with respect to a partial model assumption represented by a trail,
2. learning only non-redundant clauses out of false clauses with respect to the trail,
3. finding models in case no conflict occurs.

It is called clause learning from simple models, because the trail is restricted to ground literals.

\[
\Gamma = P(x) R(x, y) \quad \text{do not do this}
\]

\[
\Gamma = P(a) P(b) \neg R(a, b)
\]
SCL: *Simplified* Problem State

\[(\Gamma; N; U; k; D)\]

- **Γ**: Ground trail
- **N**: Initial clause set
- **U**: Learned clauses
- **k**: Level \(\xrightarrow{CDCL}\)
- **D**: State
  - \(\top\): Trail building
  - \(\bot\): \(N\) is refuted
  - A conflict clause

Initially, the state for a first-order clause set \(N\) is \((\epsilon; N; \emptyset; 0; \top)\).
SCL – quo vadis?

**ground easy case!**

\[ N = \{ P(a) \lor Q(b), \quad P(a) \lor \neg Q(b), \quad \neg P(x) \lor Q(x), \quad \neg P(x) \lor \neg Q(x) \} \]

**Decide**

\[ (\neg P(a)^7, N; \emptyset; 1; 1) \]

**Propagate**

\[ (\neg P(a) \lor Q(5), N; \emptyset; 1; 1) \]

**Resolve**

\[ (\neg P(a)^7, N; \emptyset; 1; 1; P(a) \lor Q(5); 8) \]

**End, Backtrack: \[ P(a) \]
SCL: Motivation

SCL employs a trail consisting of ground literals only:

- deciding falsity of a first-order clause with variables can be done practically efficiently and

- different ground literals don’t have common instances resulting in efficient trail operations.

- Still, non-redundant clauses with variables can be learned
  - Find falsified ground clause
  - Guide resolution on the clause level (with variables)
Resolution learns non-ground clauses

\[ N = \{ P(x) \lor Q(b), \ P(x) \lor \neg Q(y), \ \neg P(a) \lor Q(x), \ \neg P(x) \lor \neg Q(b) \} \]

Resolve \( P(x) \lor Q(b) \) with \( P(x) \lor \neg Q(y) \) \( \{ y \rightarrow b \} \)

\[ \rightarrow P(x) \lor P(x) \]

Apply factoring

\[ \rightarrow P(x) \]
SCL: *Simplified* Problem State

\[(\Gamma; N; U; k; D)\]

- \(\Gamma\): Ground trail
- \(N\): Initial clause set
- \(U\): Learned clauses
- \(k\): Level
- \(D\): State
  - \(\top\): Trail building
  - \(\bot\): \(N\) is refuted
  - A **closure** \(C\cdot\sigma\): Conflict clause \(C\) with substitution \(\sigma\)

Initially, the state for a first-order clause set \(N\) is \((\epsilon; N; \emptyset; 0; \top)\).
SimpPropagate\((\Gamma; N; U; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L_\sigma(C \lor L) \cdot \sigma; N; U; k; \top)\) provided \(C \lor L \in (N \cup U)\) technicalities missing, see later..., \((C \lor L)_\sigma\) is ground, \(C_\sigma\) is false under \(\Gamma\), and \(L_\sigma\) is undefined in \(\Gamma\).

Conflict \((\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL}} (\Gamma; N; U; \beta; k; D \cdot \sigma)\) provided \(D \in (N \cup U)\), \(D_\sigma\) false in \(\Gamma\) for a grounding substitution \(\sigma\).
Resolve

\[ (\Gamma, L\delta(\Sigma \lor L) \cdot \delta; N; U; \beta; k; (D \lor L') \cdot \sigma) \]
\[ \Rightarrow_{\text{SCL}} (\Gamma, L\delta(\Sigma \lor L) \cdot \delta; N; U; \beta; k; (D \lor C) \cdot \eta \cdot \sigma \delta) \]

provided \( L\delta = \text{comp}(L'\sigma), \eta = \text{mgu}(L, \text{comp}(L')) \)

SimpBacktrack

\[ (\Gamma \text{ comp}(L\sigma)^k; N; U; \beta; k; (D \lor L) \cdot \sigma) \]
\[ \Rightarrow_{\text{SCL}} (\Gamma'; N; U \cup \{D \lor L\}; \beta; j; \top) \]

provided \textit{a lot of technicalities...}
SCL learns non-ground clauses

\[ N = \{ P(x) \lor Q(b), \ P(x) \lor \neg Q(y), \ \neg P(a) \lor Q(x), \ \neg P(x) \lor \neg Q(b) \} \]

(\Sigma : N; \emptyset; 0; 1)

⇒ Decide (\neg P(a), N; \emptyset; 1; 0)

⇒ Propagate (\neg P(a), Q(b), P(x) \lor Q(y), N; \emptyset; 1; 0)

⇒ Consolidate (\neg P(a), Q(b), P(x) \lor Q(y), \{ x \rightarrow a, y \rightarrow b \}, N; \emptyset; 1; 0)

⇒ Rescue (\neg P(a), Q(b), P(x) \lor Q(y), N; \emptyset; 1; P(x) \lor Q(x), \{ x \rightarrow a, y \rightarrow b \}, N; \emptyset; 1; \neg P(x) \lor P(x), \{ x \rightarrow a, y \rightarrow b \})

⇒ Factorize

⇒ Skip (\neg P(a), N; \emptyset; 1; 0)

⇒ Backtrack (\Sigma : N; \emptyset; 1; 0)
Recall: CDCL soundness

2.9.10 Lemma (CDCL Soundness)

In a reasonable CDCL derivation, CDCL can only terminate in two different types of final states: \((M; N; U; k; \top)\) where \(M \models N\) and \((M; N; U; k; \bot)\) where \(N\) is unsatisfiable.
First problems in first-order

$$N = \{ P(a), \neg P(x) \lor P(f(x)) \}$$

Propagate

$$(P(a) \land 3) \land N \lor \emptyset \lor 0 \lor 1$$

Propagate

$(P(a) \land 3) \land P(f(a)) \land P(f(f(a))) \land \ldots$
- First-order Herbrand models are **infinite in general**
- In SCL:
  - **Restrict** the reasoning with respect to some ground literal \( \beta \)
  - Require that any trail literal is **smaller** than \( \beta \)
  - Use a well-founded, total, strict ordering \( \prec_\beta \) (e.g. KBO)
- Goal: Achieve termination
SCL: Problem State

\((\Gamma; N; U; \beta; k; D)\)

- \(\Gamma\): Ground trail
- \(N\): Initial clause set
- \(U\): Learned clauses
- \(\beta\): Limiting literal
- \(k\): Level
- \(D\): State
  - \(\top\): Trail building
  - \(\bot\): \(N\) is refuted
  - A closure \(C \cdot \sigma\): Conflict clause \(C\) with substitution \(\sigma\)

Initially, the state for a first-order clause set \(N\) is \((\epsilon; N; \emptyset; \beta; 0; \top)\).
SCL: Stuck states

\[ N = \{ P(a), \neg P(x) \vee P(f(x)) \} \]

set \( \beta = P(f(f(a))) \), hence exactly \( P(a) \prec_\beta \beta \) and \( P(f(a)) \prec_\beta \beta \)

"Stuck state"
Exhaustive Propagation vs. First-order logic

In propositional logic: Propagation instead of deciding is often a good idea.

In FOL: Exhaustively propagating all ground instances is a very bad idea:

\[ N' = \{ P(1, 0, x_1, \ldots, x_n), Q \lor \neg R, Q \lor R, \neg Q \lor R, \neg Q \lor \neg R \} \]

\( P(1, 0, 0, 0, \ldots, 0) \)

\( (0, 0) \)

\( (1, 1) \)

\( (0, 1, 0, 0) \)

'unsal.'
3.16.23 Example (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 $P$:

- $P(\bar{0})$
- $\neg P(\bar{1})$
- $\neg P(x, 0, \bar{1}) \lor P(x, 1, \bar{0})$

and $i$ clauses of the form

- $\neg P(x, 0, \bar{1}) \lor P(x, 1, \bar{0})$

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.
Example ctd.

\( N^4 = \{ \)

\begin{align*}
1 : & \quad P(0, 0, 0, 0) \\
2 : & \quad \neg P(x_1, x_2, x_3, 0) \lor P(x_1, x_2, x_3, 1) \\
3 : & \quad \neg P(x_1, x_2, 0, 1) \lor P(x_1, x_2, 1, 0) \\
4 : & \quad \neg P(x_1, 0, 1, 1) \lor P(x_1, 1, 0, 0) \\
5 : & \quad \neg P(0, 1, 1, 1) \lor P(1, 0, 0, 0) \\
6 : & \quad \neg P(1, 1, 1, 1)
\end{align*}

\( \}

\text{Propagate}

\text{Propagate}

\text{Propagate}

\text{Resolve}
Example ctd.

\[ Q^4 = \{ \\
1: P(0, 0, 0, 0) \\
2: \neg P(x_1, x_2, x_3, 0) \lor P(x_1, x_2, x_3, 1) \\
3: \neg P(x_1, x_2, 0, 1) \lor P(x_1, x_2, 1, 0) \\
4: \neg P(x_1, 0, 1, 1) \lor P(x_1, 1, 0, 0) \\
5: \neg P(0, 1, 1, 1) \lor P(1, 0, 0, 0) \\
6: \neg P(1, 1, 1, 1) \\
\} \]

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

Can be simulated with SCL, but not with exhaustive propagation.
Propagate \((\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L_\sigma^{(C_0 \lor L)\delta \cdot \sigma}; N; U; \beta; k; \top)\)

provided \(C \lor L \in (N \cup U)\), \(C = C_0 \lor C_1\), \(C_1\sigma = L\sigma \lor \cdots \lor L\sigma\), \(C_0\sigma\) does not contain \(L\sigma\), \(\delta\) is the mgu of the literals in \(C_1\) and \(L\), \((C \lor L)\sigma\) is ground, \((C \lor L)\sigma \prec_{\beta} \{\beta\}\), \(C_0\sigma\) is false under \(\Gamma\), and \(L\sigma\) is undefined in \(\Gamma\)

The rule Propagate applies exhaustive factoring to the propagated literal with respect to the grounding substitution \(\sigma\) and annotates the factored clause to the propagation literal on the trail.

Decide \((\Gamma; N; U; \beta; k; \top) \Rightarrow_{\text{SCL}} (\Gamma, L_\sigma^{k+1}; N; U; \beta; k + 1; \top)\)

provided \(L \in C\) for a \(C \in (N \cup U)\), \(L\sigma\) is a ground literal undefined in \(\Gamma\), and \(L_\sigma \prec_{\beta} \beta\)
Conflict \((\Gamma; N; U; \beta; k; \top) \Rightarrow_{SCL} (\Gamma; N; U; \beta; k; D \cdot \sigma)\) provided \(D \in (N \cup U)\), \(D\sigma\) false in \(\Gamma\) for a grounding substitution \(\sigma\)

These rules construct a (partial) model via the trail \(\Gamma\) for \(N \cup U\) until a conflict, i.e., a false clause with respect to \(\Gamma\) is found or all ground atoms smaller \(\beta\) are defined in \(M\) and \(M \models \text{grd}(N)^{<\beta}\).

we don't know if \(M\) is

- \(M \not\models N\)

- \(M\) is extendable to a model for \(N\)
· Guaranteed Termination
  – Only ground literals $\preceq_\beta \beta$ considered
  – There are only finitely many.
· Choosing the right $\beta$ is crucial
  – For some fragments, this gives completeness:
    Bernays-Schoenfinkel
  – In general: Every fragment with finite models

Next up: Conflict resolution rules
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 \((\Gamma, L; N; U; \beta; k; D \cdot \sigma) \Rightarrow_{SCL} (\Gamma; N; U; \beta; k - i; D \cdot \sigma)\)

provided \(\text{comp}(L)\) does not occur in \(D\sigma\), if \(L\) is a decision literal
then \(i = 1\), otherwise \(i = 0\)

Factorize \((\Gamma; N; U; \beta; k; (D \lor L \lor L') \cdot \sigma) \Rightarrow_{SCL} (\Gamma; N; U; \beta; k; (D \lor L) \eta \cdot \sigma)\)

provided \(L\sigma = L'\sigma\), \(\eta = \text{mgu}(L, L')\)
Resolve \[(\Gamma, L\delta^{(C\lor L)} \cdot \delta; N; U; \beta; k; (D \lor L') \cdot \sigma) \Rightarrow_{\text{SCL}} (\Gamma, L\delta^{(C\lor L)} \cdot \delta; N; U; \beta; k; (D \lor C) \eta \cdot \sigma \delta)\]

provided \(L\delta = \text{comp}(L'\sigma), \eta = \text{mgu}(L, \text{comp}(L'))\)

Backtrack \[(\Gamma_0, K, \Gamma_1, \text{comp}(L\sigma)^k; N; U; \beta; k; (D \lor L) \cdot \sigma) \Rightarrow_{\text{SCL}} (\Gamma_0; N; U \cup \{D \lor L\}; \beta; j; \top)\]

provided \(D\sigma\) is of level \(i' < k\), and \(\Gamma_0, K\) is the minimal trail subsequence such that there is a grounding substitution \(\tau\) with \((D \lor L)\tau\) is false in \(\Gamma_0, K\) but not in \(\Gamma_0\), and \(\Gamma_0\) is of level \(j\)

The clause \(D \lor L\) added by the rule Backtrack to \(U\) is called a learned clause.
- $\bot$ can only be derived by Resolve (or be present already in $N$)  
  $\implies$ The generation of $\bot$ is a resolution refutation
- Freedom with respect to decisions and factorizations
- Literals are not removed during resolution (eventually, Skip removes the literal from $\Gamma$)