Instance Generation (InstGen)

The idea of InstGen is to rely on a SAT solver. But instead of doing an overall grounding, a single constant is substituted for all variables and a satisfiability result of the SAT solver turned into an interpretation for the overall clause set. This interpretation either satisfies the clause st or triggers an inference via instantiation, analogous to superposition.

A *closure* is a pair clause, substitution: $C \cdot \sigma$. A substitution σ is a *proper instantiator* with respect to a literal L (clause C), if for some variable $x \in \mathrm{vars}(L)$ ($x \in \mathrm{vars}(C)$), $x\sigma$ is not a variable. Let $\succ$ be a well-founded closure ordering satifying $C \cdot \sigma \succ D \cdot \gamma$ if $C\sigma = D\gamma$ but $C\rho = D$ for some proper instantiator ρ.
More specific clauses are smaller.

The candidate model $\mathcal{I}_N$ is inductively defined over the well-founded closure ordering $\succ$ with respect to a ground model $\mathcal{I}_{N_\alpha}$ of the grounded clause set N_α. The ground clause set N_α is constructed by mapping all variables in N to a distinguished single constant α. Then if N_α is unsatisfiable, so is N. If N_α is satisfiable, N is not necessarily satisfiable and $\mathcal{I}_N$ lifts the model $\mathcal{I}_{N_\alpha}$ of N_α to a candidate model for N. Satisfiability of the clause set N_α can be more efficiently decided by a procedure for SAT, e.g., a CDCL-based SAT solver.

Similar to the model construction for superposition, suppose the sets $\delta_{D\cdot\sigma}$ have been defined for all closures $D \cdot \sigma$ smaller than $C \cdot \gamma$.

$$\mathcal{I}_{C\cdot\gamma} := \bigcup_{D\cdot\sigma \prec C\cdot\gamma} \delta_{D\cdot\sigma}$$

$$\delta_{C\cdot\gamma} := \begin{cases} \{L\gamma\} & \text{if } C\gamma \text{ is false in } \mathcal{I}_{C\cdot\gamma} \\ & C \cdot \gamma \text{ is the minimal representation of } C\gamma \text{ in } N \\ & L \in C \text{ and } L\gamma \text{ undefined in } \mathcal{I}_{C\cdot\gamma} \text{ and } L_\alpha \in \mathcal{I}_{N_\alpha} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{I}_N := \bigcup_{C\in N} \delta_{C\cdot\gamma}$$

The inference rules are:

Falsify $(N, \top) \Rightarrow_{\mathsf{IGEN}} (N, M)$

where $M = \bot$ if N_α is unsatisfiable and $M = \{L_1, \ldots, L_n\}$ if $\{L_1, \ldots, L_n\}$ is a model for N_α

Instantiate $(N \uplus \{C \vee A, D \vee \neg B\}, M) \Rightarrow_{\mathsf{IGEN}}$
$(N \uplus \{C \vee A, D \vee \neg B, (C \vee A)\sigma, (D \vee \neg B)\sigma\}, \top)$

where $M = \{L_1, \ldots, L_n\}$, $\sigma = \mathsf{mgu}(A, B)$, and σ is a proper instantiator of A or B

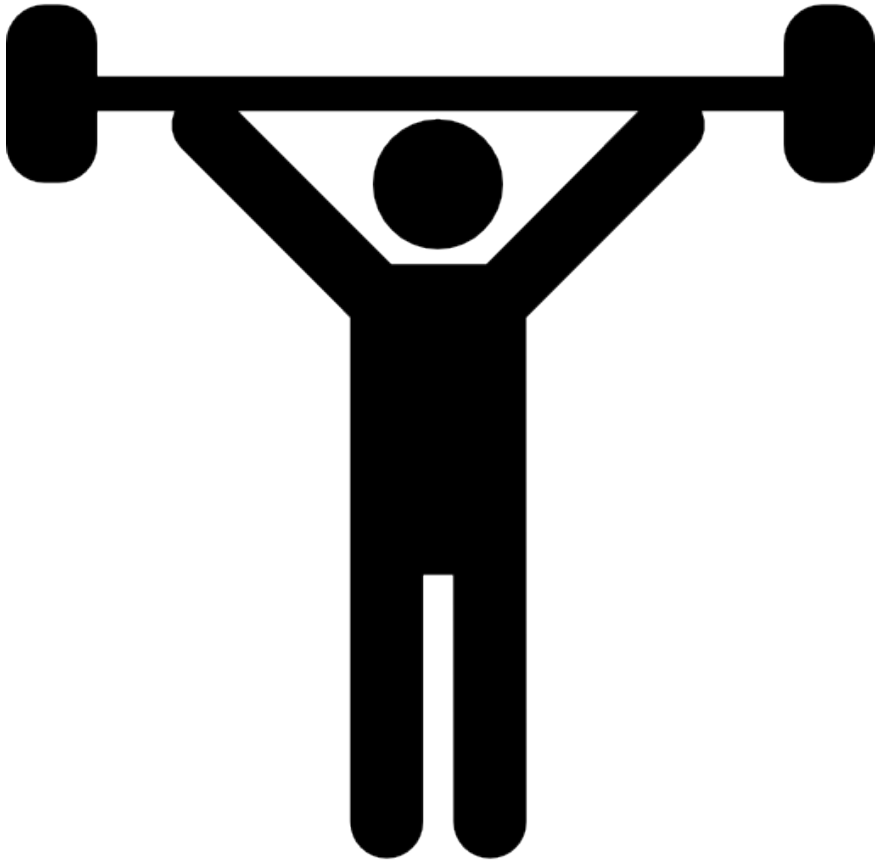
It is important that the grounding of N_α is obtained by substituting the same constant α for all variables, for otherwise the calculus becomes incomplete. For example, the two unit clauses $P(x, y); \neg P(x, x)$ are unsatisfiable. A grounding $P(a, b); \neg P(a, a)$ results in the model $\mathcal{I}_{N_\alpha} = \{P(a, b); \neg P(a, a)\}$ but Instantiate is not applicable, because the unifier $\{x \mapsto y\}$ is not a proper instantiater for both literals.
The model M is actially not used in rule Instantiate. The proof of the Theorem 3.16.4 shows that it is sufficient to consider a minimal false clause $C \vee A$ or $D \vee \neg B$ with respect to $\mathcal{I}_N$, for the inference.

3.16.4 Theorem (Completeness of InstGen)

Let $(N, \top) \Rightarrow^*_{\text{IGEN}} (N', M)$ and let (N', M) be a final state. If N is satisfiable then $M \neq \bot$ and $\mathcal{I}_{N'} \models N'$.

Redundancy can be defined analogoulsy to superposition as well. A ground closure $C \cdot \sigma$ is *redundant* in a clause set N, if there are closures $C_1 \cdot \sigma_1, \ldots, C_n \cdot \sigma_n$ from clauses $C_1, \ldots, C_n$ from N such that $C_i \cdot \sigma_i \prec C \cdot \sigma$ for all i and $C_1\sigma_1, \ldots, C_n\sigma_n \models C\sigma$. A clause C from N is *redundant* if all its ground closures $C \cdot \sigma$ are redundant.

SCL: **C**lause **L**earning from **S**imple Models



Lifting CDCL to first-order logic

CDCL – quo vadis?

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

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 wih 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.**

SCL: *Simplified* Problem State

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

- Γ: Ground trail
- N: Initial clause set
- U: Learned clauses
- k: Level
- 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?

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

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

$$N = \{P(x) \vee Q(b),\ \ P(x) \vee \neg Q(y),\ \ \neg P(a) \vee Q(x),\ \ \neg P(x) \vee \neg Q(b)\}$$

SCL: *Simplified* Problem State

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

- Γ: 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 σ

Initially, the state for a first-order clause set N is $(\epsilon; N; \emptyset; 0; \top)$.

SimpPropagate $(\Gamma; N; U; k; \top) \Rightarrow_{\mathsf{SCL}} (\Gamma, L\sigma^{(C \vee L)\cdot\sigma}; N; U; k; \top)$
provided $C \vee L \in (N \cup U)$ *technicalities missing, see later...*, $(C \vee L)\sigma$ is ground, $C\sigma$ is false under Γ, and $L\sigma$ is undefined in Γ

Conflict $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\mathsf{SCL}} (\Gamma; N; U; \beta; k; D \cdot \sigma)$
provided $D \in (N \cup U)$, $D\sigma$ false in Γ for a grounding substitution σ

Resolve $(\Gamma, L\delta^{(C \vee L)\cdot\delta}; N; U; \beta; k; (D \vee L') \cdot \sigma)$
$\Rightarrow_{\mathsf{SCL}}$ $(\Gamma, L\delta^{(C \vee L)\cdot\delta}; N; U; \beta; k; (D \vee C)\eta \cdot \sigma\delta)$
provided $L\delta = \mathsf{comp}(L'\sigma)$, $\eta = \mathsf{mgu}(L, \mathsf{comp}(L'))$

SimpBacktrack $(\Gamma\, \mathsf{comp}(L\sigma)^k; N; U; \beta; k; (D \vee L) \cdot \sigma)$
$\Rightarrow_{\mathsf{SCL}}$ $(\Gamma'; N; U \cup \{D \vee L\}; \beta; j; \top)$
provided *a lot of technicalities...*

SCL learns non-ground clauses

$$N = \{P(x) \vee Q(b),\ \ P(x) \vee \neg Q(y),\ \ \neg P(a) \vee Q(x),\ \ \neg P(x) \vee \neg Q(b)\}$$

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

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

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

SCL: Problem State

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

- Γ: Ground trail
- N: Initial clause set
- U: Learned clauses
- β: Limiting literal
- k: Level
- D: State
  - $\top$: Trail building
  - $\bot$: N is refuted
  - A **closure** $C \cdot \sigma$: Conflict clause C with substitution σ

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

SCL: Stuck states

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

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

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:

$$N' = \{P(1, 0, x_1, \ldots, x_n),\ Q \vee \neg R,\ Q \vee R,\ \neg Q \vee R,\ \neg Q \vee \neg R\}$$

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:

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

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.

Example ctd.

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

Example ctd.

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

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	$13 : \bot$

Can be simulated with SCL, but not with exhaustive propagation

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

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, $(C \vee L)\sigma \prec_\beta \{\beta\}$, $C_0\sigma$ 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.

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

provided $L \in C$ for a $C \in (N \cup U)$, $L\sigma$ is a ground literal undefined in Γ, and $L\sigma \prec_\beta \beta$

Conflict $(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\mathsf{SCL}} (\Gamma; N; U; \beta; k; D \cdot \sigma)$
provided $D \in (N \cup U)$, $D\sigma$ false in Γ for a grounding substitution σ

These rules 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 or all ground atoms smaller β are defined in M and $M \models \mathrm{grd}(N)^{\prec\beta}$.

- Guaranteed Termination
  - Only ground literals $\prec_\beta \beta$ considered
  - There are only finitely many.
- Choosing the right β 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 $(\Gamma, L; N; U; \beta; k; D \cdot \sigma) \Rightarrow_{\mathsf{SCL}}$
$(\Gamma; N; U; \beta; k - i; D \cdot \sigma)$

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

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

provided $L\sigma = L'\sigma$, $\eta = \mathsf{mgu}(L, L')$

Resolve $(\Gamma, L\delta^{(C \vee L)\cdot\delta}; N; U; \beta; k; (D \vee L') \cdot \sigma)$
$\Rightarrow_{\mathsf{SCL}}$ $(\Gamma, L\delta^{(C \vee L)\cdot\delta}; N; U; \beta; k; (D \vee C)\eta \cdot \sigma\delta)$
provided $L\delta = \mathsf{comp}(L'\sigma)$, $\eta = \mathsf{mgu}(L, \mathsf{comp}(L'))$

Backtrack $(\Gamma_0, K, \Gamma_1, \mathsf{comp}(L\sigma)^k; N; U; \beta; k; (D \vee L) \cdot \sigma)$
$\Rightarrow_{\mathsf{SCL}}$ $(\Gamma_0; N; U \cup \{D \vee L\}; \beta; j; \top)$
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, and Γ_0 is of level j

The clause $D \vee L$ 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)
  $\Longrightarrow$ 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 Γ)

3.16.8 Definition (Sound States)

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

1. Γ is a consistent sequence of annotated ground literals, i.e. for a ground literal L it cannot be that $L \in \Gamma$ and $\neg L \in \Gamma$
2. for each decomposition $\Gamma = \Gamma_1, L\sigma^{C \vee L \cdot \sigma}, \Gamma_2$ 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 $\Gamma = \Gamma_1, L^k, \Gamma_2$ we have that L is undefined in Γ_1,
4. $N \models U$,
5. if $D = C \cdot \sigma$ then $C\sigma$ is false under Γ and $N \models C$. In particular, $\mathrm{grd}^{\prec_\beta \beta}(N) \models C\sigma$,
6. for any $L \in \Gamma$ we have $L \prec_\beta \beta$ and there is a $C \in N \cup U$ such that $L \in C$.

3.16.9 Lemma (Soundness of the initial state)

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

Proof.

Criteria 1–3 and 6 are trivially satisfied by $\Gamma = \epsilon$. Furthermore, $N \models \emptyset$, fulfilling criterion 4. Lastly, criterion 5 is trivially fulfilled for $D = \top$. □

3.16.10 Theorem (Soundness of SCL)

All SCL rules preserve soundness, i.e. they map a sound state onto a sound state.

3.16.9 Lemma (Soundness of the initial state)

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

Proof.

Criteria 1–3 and 6 are trivially satisfied by $\Gamma = \epsilon$. Furthermore, $N \models \emptyset$, fulfilling criterion 4. Lastly, criterion 5 is trivially fulfilled for $D = \top$. □

3.16.10 Theorem (Soundness of SCL)

All SCL rules preserve soundness, i.e. they map a sound state onto a sound state.

Corollary (Sondness of SCL)

The rules of SCL are sound, hence SCL starting with an initial state is sound.

Proof.

Follows by induction over the size of the run. The base case is handled by Lemma 3.16.9, the induction step is contained in Theorem 3.16.10. □

3.16.12 Definition (Reasonable Runs)

A sequence of SCL rule applications is called a *reasonable run* if the rule Decide does not enable an immediate application of rule Conflict.

3.16.13 Definition (Regular Runs)

A sequence of SCL rule applications is called a *regular run* if it is a reasonable run and the rule Conflict has precedence over all other rules.

3.16.14 Theorem (Correct Termination)

If in a regular run no rules are applicable to a state $(\Gamma; N; U; \beta; k; D)$ then either $D = \bot$ and N is unsatisfiable or $D = \top$ and $\mathrm{grd}(N)^{\prec_\beta \beta}$ is satisfiable and $\Gamma \models \mathrm{grd}(N)^{\prec_\beta \beta}$.

3.16.14 Theorem (Correct Termination)

If in a regular run no rules are applicable to a state $(\Gamma; N; U; \beta; k; D)$ then either $D = \bot$ and N is unsatisfiable or $D = \top$ and $\mathrm{grd}(N)^{\prec_\beta \beta}$ is satisfiable and $\Gamma \models \mathrm{grd}(N)^{\prec_\beta \beta}$.

3.16.15 Lemma (Resolve in regular runs)

Consider the derivation of a conflict state $(\Gamma, L; N; U; \beta; k; \top) \Rightarrow_{\text{Conflict}} (\Gamma, L; N; U; \beta; k; D)$. In a regular run, during conflict resolution L is not a decision literal and at least the literal L is resolved.

3.16.15 Lemma (Resolve in regular runs)

Consider the derivation of a conflict state $(\Gamma, L; N; U; \beta; k; \top) \Rightarrow_{\text{Conflict}} (\Gamma, L; N; U; \beta; k; D)$. In a regular run, during conflict resolution L is not a decision literal and at least the literal L is resolved.

3.16.16 Definition (State Induced Ordering)

Let $(L_1, L_2, \ldots, L_n; N; U; \beta; k; D)$ be a sound state of SCL. The trail induces a total well-founded strict order on the defined literals by

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

We extend $\prec_\Gamma$ to a strict total order on all literals where all undefined literals are larger than $\mathrm{comp}(L_n)$. 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 $\prec_\Gamma$.

3.16.17 Theorem (Learned Clauses in Regular Runs)

Let $(\Gamma; N; U; \beta; k; C_0 \cdot \sigma_0)$ be the state resulting from the application of Conflict 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 $N \cup U$ and $\prec_\Gamma$.

- During a run, the ordering of literals changes
- Hence, $\prec_\Gamma$ changes as well!
- Non-redundancy property of Theorem 3.16.17 reflects state at time of creation of learned clause
- At time of creation, no need to check for redundancy
- Still, **all** $\prec_\Gamma$ **contain** the fixed clause subset ordering $\prec_\subseteq$

3.16.19 Theorem (Termination)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ terminates.

Lemma (Termination without Backtrack)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ that does not use the Backtrack rule terminates.

$$\mathcal{M}(\Gamma, N; U; \beta; k; \top) = (1, \quad |\{P \mid P \prec_B \beta\}| - |\Gamma|, \quad 0 \quad)$$
$$\mathcal{M}(\Gamma, N; U; \beta; k; C) = (0, \quad \#\text{possible resolutions}, \quad |C| \quad)$$

3.16.19 Theorem (Termination)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ terminates.

Lemma (Termination without Backtrack)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ that does not use the Backtrack rule terminates.

$$\mathcal{M}(\Gamma, N; U; \beta; k; \top) = (1, \quad |\{P \mid P \prec_B \beta\}| - |\Gamma|, \quad 0 \quad)$$
$$\mathcal{M}(\Gamma, N; U; \beta; k; C) = (0, \quad \#\text{possible resolutions}, \quad |C| \quad)$$

3.16.19 Theorem (Termination)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ terminates.

Lemma (Termination without Backtrack)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ that does not use the Backtrack rule terminates.

$$\mathcal{M}(\Gamma, N; U; \beta; k; \top) = (1, \quad |\{P \mid P \prec_B \beta\}| - |\Gamma|, \quad 0 \quad)$$
$$\mathcal{M}(\Gamma, N; U; \beta; k; C) = (0, \quad \text{\#possible resolutions}, \quad |C| \quad)$$

Lemma (Termination with Backtrack)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ cannot use the Backtrack rule infinitely often.

Proof.

Firstly, for a regular run, by Theorem 3.16.17, all learned clauses are non-redundant under $\prec_\Gamma$. Those clauses are also non-redundant under the fixed subset ordering $\prec_\subseteq$, which is well-founded. Due to the restriction of all clauses to be smaller than $\{\beta\}$, the overall number of non-redundant ground clauses is finite. So Backtrack can only be invoked finitely many times. □

Lemma (Termination with Backtrack)

Any regular run of $\Rightarrow_{\mathsf{SCL}}$ cannot use the Backtrack rule infinitely often.

Proof.

Firstly, for a regular run, by Theorem 3.16.17, all learned clauses are non-redundant under $\prec_\Gamma$. Those clauses are also non-redundant under the fixed subset ordering $\prec_\subseteq$, which is well-founded. Due to the restriction of all clauses to be smaller than $\{\beta\}$, the overall number of non-redundant ground clauses is finite. So Backtrack can only be invoked finitely many times. □

3.16.20 Theorem (SCL Refutational Completeness)

If N is unsatisfiable, such that some finite $N' \subseteq \mathrm{grd}(N)$ is unsatisfiable and β is $\prec_\beta$ larger than all literals in N' then any regular run from $(\epsilon; N; \emptyset; \beta; 0; \top)$ of SCL derives $\bot$.

Proof.

By Theorem 3.16.19 and Theorem 3.16.14. □

3.16.18 Theorem (BS Non-Redundancy is NEXPTIME-Complete)

Deciding non-redundancy of a BS clause C with respect to a finite BS clause set $N^{\preceq C}$ is NEXPTIME-Complete.

Obviously, given some unsatisfiable clause set N there is no way to efiiciently compute some β such that $\mathit{ground}(N)^{\prec_\beta}$ is unsatisfiable. Therefore, in an implementation, the below rule Grow is needed to eventually provide a semi-decision procedure.

Grow $$(\Gamma; N; U; \beta; k; \top) \Rightarrow_{\mathsf{SCL}} (\epsilon; N; U; \beta'; 0; \top)$$

provided $\Gamma \models \mathrm{grd}(N)^{\prec_\beta}$ and $\beta \prec_\beta \beta'$

3.16.21 Theorem (SCL decides the BS fragment)

SCL restricted to regular runs decides satisfiability of a BS clause set if β is set appropriately.

Proof.

Let B be the set of constants in the BS clause set N. Then define $\prec_\beta$ and β such that $L \prec_\beta \beta$ for all $L \in \text{grd}^{\prec_\beta \beta}(N)$. Following the proof of Theorem 3.16.19, any SCL regular run will terminate on a BS clause set. □

The End (of SCL)