Craig-Interpolation

A theoretical application of superposition is Craig-Interpolation:

Theorem 3.38 (Craig 1957) *Let ϕ and ψ be two propositional formulas such that $\phi \models \psi$. Then there exists a formula χ (called the* interpolant *for $\phi \models \psi$), such that χ contains only prop. variables occurring both in ϕ and in ψ, and such that $\phi \models \chi$ and $\chi \models \psi$.*

Proof. Translate ϕ and $\neg\psi$ into CNF. let N and M, resp., denote the resulting clause set. Choose an atom ordering $\succ$ for which the prop. variables that occur in ϕ but not in ψ are maximal. Saturate N into N^* w. r. t. $Sup^{\succ}_{\text{sel}}$ with an empty selection function sel . Then saturate $N^* \cup M$ w. r. t. $Sup^{\succ}_{\text{sel}}$ to derive $\bot$. As N^* is already saturated, due to the ordering restrictions only inferences need to be considered where premises, if they are from N^*, only contain symbols that also occur in ψ. The conjunction of these premises is an interpolant χ. The theorem also holds for first-order formulas. For universal formulas the above proof can be easily extended. In the general case, a proof based on superposition technology is more complicated because of Skolemization. □

Redundancy

So far: local restrictions of the resolution inference rules using orderings and selection functions.

Is it also possible to delete clauses altogether? Under which circumstances are clauses unnecessary? (Conjecture: e. g., if they are tautologies or if they are subsumed by other clauses.)

Intuition: If a clause is guaranteed to be neither a minimal counterexample nor productive, then we do not need it.

A Formal Notion of Redundancy

Recall: Let N be a set of ground clauses and C a ground clause (not necessarily in N). C is called *redundant* w. r. t. N, if there exist $C_1, \ldots, C_n \in N$, $n \geq 0$, such that $C_i \prec C$ and $C_1, \ldots, C_n \models C$.

Redundancy for general clauses: C is called *redundant* w. r. t. N, if all ground instances $C\sigma$ of C are redundant w. r. t. $G_\Sigma(N)$.

Note: The same ordering $\prec$ is used for ordering restrictions and for redundancy (and for the completeness proof).

Examples of Redundancy

Proposition 3.39 *Recall the redundancy criteria:*

- C *tautology (i. e.,* $\models C$*)* $\Rightarrow$ C *redundant w. r. t. any set* N*.*
  Tautology Deletion
- $C\sigma \subset D \Rightarrow D$ *redundant w. r. t.* $N \cup \{C\}$*.*
  Subsumption
- $C\sigma \subseteq D \Rightarrow D \vee \overline{L}\sigma$ *redundant w. r. t.* $N \cup \{C \vee L,\ D\}$*.*
  Subsumption Resolution

Saturation up to Redundancy

N is called *saturated up to redundancy* (w. r. t. $Sup^{\succ}_{\text{sel}}$)

$$:\Leftrightarrow Sup^{\succ}_{\text{sel}}(N \setminus Red(N)) \subseteq N \cup Red(N)$$

Theorem 3.40 *Let* N *be saturated up to redundancy. Then*

$$N \models \bot \Leftrightarrow \bot \in N$$

Proof (Sketch). (i) Ground case:

- consider the construction of the candidate interpretation $N^{\succ}_{\mathcal{I}}$ for $Sup^{\succ}_{\text{sel}}$
- redundant clauses are not productive
- redundant clauses in N are not minimal counterexamples for $N^{\succ}_{\mathcal{I}}$

The premises of "essential" inferences are either minimal counterexamples or productive.

(ii) Lifting: no additional problems over the proof of Theorem 3.37. □

Monotonicity Properties of Redundancy

Theorem 3.41

(i) $N \subseteq M \Rightarrow Red(N) \subseteq Red(M)$

(ii) $M \subseteq Red(N) \Rightarrow Red(N) \subseteq Red(N \setminus M)$

We conclude that redundancy is preserved when, during a theorem proving process, one adds (derives) new clauses or deletes redundant clauses. Recall that $Red(N)$ may include clauses that are not in N.

A First-Order Superposition Theorem Prover

Straightfotward extension of the propositional STP prover.

3 clause sets:

N(ew) containing new inferred clauses
U(sable) containing reduced new inferred clauses
clauses get into *W(orked) O(ff)* once their inferences have been computed

Strategy:

Inferences will only be computed when there are no possibilities for simplification

Rewrite Rules for STP

Tautology Deletion
$(N \uplus \{C\}; U; WO) \quad \Rightarrow_{STP} \quad (N; U; WO)$

if C is a tautology

Forward Subsumption
$(N \uplus \{C\}; U; WO) \quad \Rightarrow_{STP} \quad (N; U; WO)$

if some $D \in (U \cup WO)$ subsumes C, $D\sigma \subseteq C$

Backward Subsumption U
$(N \uplus \{C\}; U \uplus \{D\}; WO) \quad \Rightarrow_{STP} \quad (N \cup \{C\}; U; WO)$

if C strictly subsumes D ($C\sigma \subset D$)

Backward Subsumption WO
$(N \uplus \{C\}; U; WO \uplus \{D\}) \quad \Rightarrow_{STP} \quad (N \cup \{C\}; U; WO)$

if C strictly subsumes D ($C\sigma \subset D$)

Forward Subsumption Resolution
$(N \uplus \{C_1 \vee L\}; U; WO) \quad \Rightarrow_{STP} \quad (N \cup \{C_1\}; U; WO)$

if $C_2 \vee L' \in (U \cup WO)$ such that $C_2\sigma \subseteq C_1$ and $L'\sigma = \overline{L}$

Backward Subsumption Resolution U
$(N \uplus \{C_1 \vee L\}; U \uplus \{C_2 \vee L'\}; WO) \quad \Rightarrow_{STP} \quad (N \cup \{C_1 \vee L\}; U \uplus \{C_2\}; WO)$

if $C_1\sigma \subseteq C_2$ and $L'\sigma = \overline{L}$

Backward Subsumption Resolution WO
$(N \uplus \{C_1 \vee L'\}; U; WO \uplus \{C_2 \vee L\}) \quad \Rightarrow_{STP} \quad (N \cup \{C_1 \vee L\}; U; WO \uplus \{C_2\})$

if $C_1\sigma \subseteq C_2$ and $L'\sigma = \overline{L}$

Clause Processing

$(N \uplus \{C\}; U; WO) \quad \Rightarrow_{STP} \quad (N; U \cup \{C\}; WO)$

Inference Computation

$(\emptyset; U \uplus \{C\}; WO) \quad \Rightarrow_{STP} \quad (N; U; WO \cup \{C\})$

where N is the set of clauses derived by first-order superposition inferences from C and clauses in WO.

Implementation

Although first-order and propositional subsumption just differ in the matcher σ, propositional subsumption between two clauses C and D can be decided in $O(n)$, $n = |C| + |D|$ whereas first-order subsumption is NP-complete.

Hyperresolution

There are *many* variants of resolution. (We refer to [Bachmair, Ganzinger: Resolution Theorem Proving] for further reading.)

One well-known example is hyperresolution (Robinson 1965):

> Assume that several negative literals are selected in a clause C. If we perform an inference with C, then one of the selected literals is eliminated.
>
> Suppose that the remaining selected literals of C are again selected in the conclusion. Then we must eliminate the remaining selected literals one by one by further resolution steps.

Hyperresolution replaces these successive steps by a single inference. As for $Sup^{\succ}_{\text{sel}}$, the calculus is parameterized by an atom ordering $\succ$ and a selection function sel.

$$\frac{D_1 \vee B_1 \quad \ldots \quad D_n \vee B_n \qquad C \vee \neg A_1 \vee \ldots \vee \neg A_n}{(D_1 \vee \ldots \vee D_n \vee C)\sigma}$$

with $\sigma = \text{mgu}(A_1 \doteq B_1, \ldots, A_n \doteq B_n)$, if

(i) $B_i\sigma$ strictly maximal in $D_i\sigma$, $1 \leq i \leq n$;

(ii) nothing is selected in D_i;

(iii) the indicated occurrences of the $\neg A_i$ are exactly the ones selected by sel, or else nothing is selected in the right premise and $n = 1$ and $\neg A_1\sigma$ is maximal in $C\sigma$.

Similarly to superposition (resolution), hyperresolution has to be complemented by a factorization inference.

As we have seen, hyperresolution can be simulated by iterated binary superposition.

However this yields intermediate clauses which HR might not derive, and many of them might not be extendable into a full HR inference.

3.12 Summary: Superposition Theorem Proving

- Superposition is a machine calculus.
- Subtle interleaving of enumerating instances and proving inconsistency through the use of unification.
- Parameters: atom ordering $\succ$ and selection function sel. On the non-ground level, ordering constraints can (only) be solved approximatively.
- Completeness proof by constructing candidate interpretations from productive clauses $C \vee A$, $A \succ C$; inferences with those reduce counterexamples.
- *Local* restrictions of inferences via $\succ$ and sel
  $\Rightarrow$ fewer proof variants.
- *Global* restrictions of the search space via elimination of redundancy
  $\Rightarrow$ computing with "smaller" clause sets;
  $\Rightarrow$ termination on many decidable fragments.
- However: not good enough for dealing with orderings, equality and more specific algebraic theories (lattices, abelian groups, rings, fields) or arithmetic
  $\Rightarrow$ further specialization of inference systems required.

Other Inference Systems

- Tableaux
- Instantiation-based methods
  Resolution-based instance generation
  Disconnection calculus
  ...
- Natural deduction
- Sequent calculus/Gentzen calculus
- Hilbert calculus

One major problem with all those calculi concerning automation is that they contain a rule either guessing instances or limiting the use of formulas. So the procedure has to guess instances and/or the number of copies of formulas. For example rules like:

Universal Quantification
$S \cup \{\forall x\ \phi\} \quad \Rightarrow \quad S \cup \{\forall x\ \phi\} \cup \phi\{x \mapsto t\}$

for some ground term $t \in \mathrm{T}_\Sigma$

Existential Quantification
$S \cup \{\exists x\ \phi\} \quad \Rightarrow \quad S \cup \{\exists x\ \phi\} \cup \phi\{x \mapsto a\}$

for some constant a new to ϕ

4 First-Order Logic with Equality

Equality is the most important relation in mathematics and functional programming.

In principle, problems in first-order logic with equality can be handled by any prover for first-order logic without equality:

4.1 Handling Equality Naively

Proposition 4.1 *Let F be a closed first-order formula with equality. Let $\sim \notin \Pi$ be a new predicate symbol. The set $Eq(\Sigma)$ contains the formulas*

$$\forall x\,(x \sim x)$$
$$\forall x, y\,(x \sim y \rightarrow y \sim x)$$
$$\forall x, y, z\,(x \sim y \wedge y \sim z \rightarrow x \sim z)$$
$$\forall \vec{x}, \vec{y}\,(x_1 \sim y_1 \wedge \cdots \wedge x_n \sim y_n \rightarrow f(x_1, \ldots, x_n) \sim f(y_1, \ldots, y_n))$$
$$\forall \vec{x}, \vec{y}\,(x_1 \sim y_1 \wedge \cdots \wedge x_m \sim y_m \wedge p(x_1, \ldots, x_m) \rightarrow p(y_1, \ldots, y_m))$$