Hausdorff School: "Formal Mathematics and Computer-Assisted Proving"

Automated Theorem Proving with Ordered Resolution and Superposition

Alexander Bentkamp

These notes are partly based on Uwe Waldmann's lecture notes:
https://rg1-teaching.mpi-inf.mpg.de/autrea-ws21/readings.html

1 Motivation

Automatic theorem provers are computer programs that prove mathematical theorems fully automatically. The user only provides the conjecture they would like to prove. With a few notable exceptions[1], they are far from the capabilities of human mathematicians. An exciting application however, is to integrate automatic theorem provers into proof assistants. Lean does not yet have such an integration, but various other proof assistants do. They are called "hammers" in honor of Sledgehammer, possibly the most successful such integration.

Automatic provers work best when their input is not too large. So instead of providing the proof assistant's entire library to the automatic prover, hammers first determine the most promising facts from the library that might be needed to prove a given conjecture.

The second step is to translate the selected facts and the conjecture. Proof assistants typically operate on richer logics (dependent type theory or higher-order logic) than automatic provers (typically first-order logic).

The automatic provers then attempt to prove the translated problem. If a proof is found, the proof is finally reconstructed within the proof assistant, where it is independently verified.

The two most successful automated theorem proving technologies in this context are SMT and superposition. In this lecture series, we will focus on the ordered resolution calculus, which is the restriction of superposition to first-order logic without equality. My goal is to show that automated theorem provers can not only help us to prove mathematical theorems, but that studying the calculi implemented in these provers is also mathematically interesting.

2 First-Order Logic

Syntax We fix a set Ω of function symbols, and a set Π of predicate symbols. Each of these symbols is associated with an arity $\mathrm{arr} : \Omega \cup \Pi \to \mathbb{N}_{\geq 0}$. We call

[1] https://en.wikipedia.org/wiki/Computer-assisted_proof#Applications

$\Sigma = (\Omega, \Pi, \mathrm{arr})$ the signature. Moreover, we fix a countably infinite set $\mathcal{V}$ of variables.

A *term* is inductively defined as follows:

- Every variable $x \in \mathcal{V}$ is a term.
- $\mathsf{f}(t_1, \ldots, t_n)$ is a term for $\mathsf{f} \in \Omega$ with $\mathrm{arr}(\mathsf{f}) = n$ and terms t_1, ..., t_n. (If $n = 0$, we write f for $\mathsf{f}()$)

A *formula* is inductively defined as follows:

- $\mathsf{p}(t_1, \ldots, t_n)$ is a formula (called *atom*) for $\mathsf{p} \in \Pi$ with $\mathrm{arr}(\mathsf{p}) = n$ and terms t_1, ..., t_n. (If $n = 0$, we write p for $\mathsf{p}()$)
- The expressions $\top$, $\bot$, $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$, $\forall x.\, \varphi$, and $\exists x.\, \varphi$ are formulas if φ and ψ are formulas.

A term or formula is called *ground* if it contains no variables.

There are two variants of FOL: with and without equality. In FOL with $\approx$, we require that Π contains a predicate symbol $\approx$, and we write $t \approx s$ for $\approx(s, t)$.

Semantics An *interpretation* $\mathcal{I} = (\mathcal{U}, \mathcal{J})$ is a pair, consisting of a nonempty set $\mathcal{U}$ and a function $\mathcal{J}$, which associates with each $\mathsf{f} \in \Omega$ a function $\mathcal{J}(\mathsf{f}) : \mathcal{U}^{\mathrm{arr}(f)} \to \mathcal{U}$ and with each $\mathsf{p} \in \Pi$ a set $\mathcal{J}(\mathsf{p}) \subseteq \mathcal{U}^{\mathrm{arr}(p)}$.

In FOL with $\approx$, we require that $\mathcal{J}(\approx) = \{(a, a) \mid a \in \mathcal{U}\}$.

A *valuation* is a function $\xi : \mathcal{V} \to \mathcal{U}$. For an interpretation $(\mathcal{U}, \mathcal{J})$ and a valuation ξ, the denotation of a term is recursively defined as

- $[\![x]\!]^\xi_\mathcal{J} = \xi(x)$ for $x \in \mathcal{V}$
- $[\![\mathsf{f}(t_1, \ldots, t_n)]\!]^\xi_\mathcal{J} = \mathcal{J}(\mathsf{f})([\![t_1]\!]^\xi_\mathcal{J}, \ldots, [\![t_n]\!]^\xi_\mathcal{J})$ for $\mathsf{f} \in \Omega$ and terms $t_1, \ldots, t_n$.

For ground terms t, we also write $[\![t]\!]_\mathcal{J}$ for $[\![t]\!]^\xi_\mathcal{J}$ because it does not depend on ξ.

A formula φ being *true* under $\mathcal{I}$ and ξ, written $\mathcal{I}, \xi \vDash \varphi$ is recursively defined as:

$$
\begin{aligned}
\mathcal{I}, \xi \vDash \mathsf{p}(t_1, ..., t_n) &\quad \text{iff } ([\![t_1]\!]^\xi_\mathcal{J}, ..., [\![t_n]\!]^\xi_\mathcal{J}) \in \mathcal{J}(\mathsf{p}) \\
\mathcal{I}, \xi \vDash \top &\quad \text{always holds} \\
\mathcal{I}, \xi \vDash \bot &\quad \text{never holds} \\
\mathcal{I}, \xi \vDash \neg\varphi &\quad \text{iff } \mathcal{I}, \xi \nvDash \varphi \\
\mathcal{I}, \xi \vDash \varphi \wedge \psi &\quad \text{iff } \mathcal{I}, \xi \vDash \varphi \text{ and } \mathcal{I}, \xi \vDash \psi \\
\mathcal{I}, \xi \vDash \varphi \vee \psi &\quad \text{iff } \mathcal{I}, \xi \vDash \varphi \text{ or } \mathcal{I}, \xi \vDash \psi \\
\mathcal{I}, \xi \vDash \varphi \to \psi &\quad \text{iff } \mathcal{I}, \xi \nvDash \varphi \text{ or } \mathcal{I}, \xi \vDash \psi \\
\mathcal{I}, \xi \vDash \forall x.\, \varphi &\quad \text{iff for every } a \in \mathcal{U},\ \mathcal{I}, \xi[x \mapsto a] \vDash \varphi \\
\mathcal{I}, \xi \vDash \exists x.\, \varphi &\quad \text{iff for some } a \in \mathcal{U},\ \mathcal{I}, \xi[x \mapsto a] \vDash \varphi
\end{aligned}
$$

where

$$
\xi[x \mapsto a](y) = \begin{cases} a \text{ if } y = x \\ \xi(y) \text{ otherwise} \end{cases}
$$

- $\mathcal{I}$ is a *model* of φ, written $\mathcal{I} \models \varphi$, iff $\mathcal{I}, \xi \models \varphi$ for all valuations ξ.
- A formula is *valid* if $\mathcal{I} \models \varphi$ for all interpretations $\mathcal{I}$.
- A formula is *satisfiable* if it has a model.
- A set of formulas N *entails* a set of formulas M, written $N \models M$, if every model of all formulas in N is also a model of all formulas in M.

Example Let $\Omega = \{\mathsf{f}\}$ with $\mathrm{arr}(\mathsf{f}) = 2$ and $\Pi = \{\mathsf{p}\}$ with $\mathrm{arr}(\mathsf{p}) = 2$.

- The formula $\varphi = \forall x.\, \forall y.\, \mathsf{p}(x, \mathsf{f}(x, y))$ is satisfiable:
  - For example, $(\mathcal{U}, \mathcal{J})$ with $\mathcal{U} = \mathbb{N}$, $\mathcal{J}(\mathsf{f}) = +$, and $\mathcal{J}(\mathsf{p}) = \{(a, b) \mid a \leq b\}$ is a model:

$$\begin{aligned} & \mathcal{I} \models \varphi \\ \text{iff } & \mathcal{I}, \xi \models \varphi \text{ for all } \xi \\ \text{iff } & \mathcal{I}, \xi[x \mapsto a][y \mapsto b] \models \mathsf{p}(x, \mathsf{f}(x, y)) \text{ for all } a, b \in \mathcal{U} \\ \text{iff } & (a, \mathcal{J}(\mathsf{f})(a, b)) \in \mathcal{J}(\mathsf{p}) \\ \text{iff } & a \leq a + b \end{aligned}$$

  - Actually, any interpretation with $\mathcal{J}(\mathsf{p}) = \mathcal{U} \times \mathcal{U}$ is a model as well.
- The formula $\varphi = \exists x.\, \mathsf{p}(x, x) \wedge \neg \mathsf{p}(x, x)$ is unsatisfiable because we cannot have an $a \in \mathcal{U}$ such that $(a, a) \in \mathcal{J}(\mathsf{p})$ and $(a, a) \notin \mathcal{J}(\mathsf{p})$.

3 Herbrand Interpretations

Definition 1. Let $(\Omega, \Pi, \mathrm{arr})$ be a signature for FOL without $\approx$. Let Ω contain at least one symbol of arity 0. A *Herbrand interpretation* is an interpretation $\mathcal{I} = (\mathcal{U}, \mathcal{J})$ where $\mathcal{U}$ is the set of all ground terms over $(\Omega, \Pi, \mathrm{arr})$ and $\mathcal{J}(\mathsf{f})(t_1, ..., t_n) = \mathsf{f}(t_1, ..., t_n)$ for all function symbols $\mathsf{f} \in \Omega$.

(We can always add a dummy symbol of arity 0; this does not change the satisfiablity of formulas.)

A Herbrand interpretation thus fixes the universe $\mathcal{U}$ and the interpretation of function symbols $\mathcal{J}(\mathsf{f})$. But the interpretation of predicate symbols $\mathcal{J}(\mathsf{p})$ can be chosen freely:

Lemma 2. *For a given signature, there is a bijection between sets S of ground atoms and Herbrand interpretations $\mathcal{I}$ defined by*

$$\mathsf{p}(t_1, ..., t_n) \in S \text{ iff } \mathcal{I} \models \mathsf{p}(t_1, ..., t_n)$$

Below, we will identify sets of ground atoms with their corresponding Herbrand interpretation.

4 Proving by Saturation

Paradigm for automated reasoning, foundation of both resolution and superposition

- Given: prove formula φ
- Negate: $\neg\varphi$
- Bring $\neg\varphi$ into a *normal form*, resulting in a set of formulas N ("essentially equivalent" to $\neg\varphi$)
- Saturation loop:
  - Add formulas to N that follow from N according to *inference rules*
  - Remove redundant formulas according to *redundancy criterion*
- Stop when $\bot$ has been derived
- Give up when no more inference rules apply

5 Clausal Normal Form

We use the clausal normal form (CNF). CNF formulas are structured as follows:

- An *atom* is a formula of the form $\mathsf{p}(t_1, \ldots, t_n)$.
- A *literal* is a formula of the form A or $\neg A$ for some atom A.
- A *clause* is a formula of the form $L_1 \vee \cdots \vee L_n$ (or the empty clause $\bot$) for some literals L_i.

A CNF is a set of clauses, where

- we understand all variables as universally quantified;
- we understand the clauses in the set to be connected by '$\wedge$';

Moreover, we will view clauses as multisets of literals—i.e.,

- We do not care about order of literals: $L_1 \vee L_2 \vee L_3 = L_2 \vee L_3 \vee L_1$.
- We care about multiplicity: $L \vee L \neq L$.

A CNF of a formula can be obtained as follows:

- Eliminate implications: replace $\varphi \to \psi$ by $\neg\varphi \vee \psi$
- Pull $\forall$ and $\exists$ outwards
- Replace $\exists$ with Skolem function symbols

- Push negations inwards using DeMorgan's laws and eliminate double negations.
- Distribute $\vee$ inwards over $\wedge$
- Eliminate $\top$ and $\bot$
- Omit $\forall$s and view each conjunct as a clause

The result of this process is satisfiable iff the original formula was satisfiable. So if the resulting set of clauses implies $\bot$, then the original clause set implies $\bot$.

Example Let p be a binary predicate symbol and c a nullary function symbol.

$$\begin{aligned}
&\neg((\forall x.\, \exists y.\, \mathsf{p}(x,y)) \to \exists z.\, \mathsf{p}(\mathsf{c},z)) \\
\text{Eliminate implications:}\quad &\neg(\neg(\forall x.\, \exists y.\, \mathsf{p}(x,y)) \vee \exists z.\, \mathsf{p}(\mathsf{c},z)) \\
\text{Pull } \forall \text{ and } \exists \text{ outwards:}\quad &\forall x.\, \exists y.\, \forall z.\, \neg(\neg\mathsf{p}(x,y) \vee \mathsf{p}(\mathsf{c},z)) \\
\text{Replace } \exists \text{ with Skolem function symbols:}\quad &\forall x.\, \forall z.\, \neg(\neg\mathsf{p}(x,\mathsf{f}(x)) \vee \mathsf{p}(\mathsf{c},z)) \\
\text{Push negations inwards:}\quad &\forall x.\, \forall z.\, \mathsf{p}(x,\mathsf{f}(x)) \wedge \neg\mathsf{p}(\mathsf{c},z) \\
\text{Resulting clauses:}\quad &\mathsf{p}(x,\mathsf{f}(x)),\ \neg\mathsf{p}(\mathsf{c},z)
\end{aligned}$$

6 Inference Systems

A (clausal) inference is a tuple of clauses

$$(C_1, ..., C_n, C_{n+1}), \quad n \geq 0$$

written

$$\frac{\overbrace{C_1 \quad ... \quad C_n}^{\text{premises}}}{\underbrace{C_{n+1}}_{\text{conclusion}}}$$

An inference system is a set of such inferences.

Let Γ be an inference system.

- A clause C is *derivable* by Γ from a clause set N, written $N \vdash_\Gamma C$, if $C \in N$ or if C is the conclusion of a Γ-inference whose premises are derivable by Γ from N.
- Γ is *sound* if $N \vdash_\Gamma C$ implies $N \models C$ for all C and N (or in other words: for every inference the premises entail the conclusion).
- Γ is *complete* if $N \models C$ implies $N \vdash_\Gamma C$ for all C and N
- Γ is *refutationally complete* if $N \models \bot$ implies $N \vdash_\Gamma \bot$

- A set N of clauses is *saturated* w.r.t. an inference system Γ if the conclusion of any Γ-inference from clauses in N is contained in N.

Lemma 3. *An inference system Γ is refutationally complete if and only if all clause sets N saturated w.r.t. Γ with $\bot \notin N$ are satisfiable.*

Proof. "$\Leftarrow$": We assume that all saturated N with $\bot \notin N$ are satisfiable. Let $N_0 \models \bot$ (i.e., N_0 is unsatisfiable). To show: $N_0 \vdash_\Gamma \bot$. Let $N_\infty = \{C \mid N_0 \vdash_\Gamma C\}$. Clearly, N_∞ is saturated. Moreover, since $N_0 \subseteq N_\infty$, N_∞ is unsatisfiable. By assumption all saturated N with $\bot \notin N$ are satisfiable, so we must have $\bot \in N_\infty$ and thus $N_0 \vdash_\Gamma \bot$.

"$\Rightarrow$": We assume that Γ is refutationally complete. Let N be saturated and $\bot \notin N$. It follows that $N \nvdash_\Gamma \bot$. By refutational completeness, $N \not\models \bot$ and thus N is satisfiable. □

7 Resolution

Resolution is an inference system for FOL with $\approx$. History:

- Davis and Putnam, 1960: Resolution, but no efficient treatment of variables
- Robinson, 1965: Resolution with unification
- Slagle, 1967 / Maslov, 1968: Ordered resolution
- Bachmair and Ganzinger, 1990: Resolution with redundancy

7.1 Ground resolution

For now, we ignore variables and assume that all clauses are ground.

The ground resolution calculus has two inference rules: resolution and factoring

$$\frac{D \vee A \qquad C \vee \neg A}{D \vee C}\,\textsc{Res}_{\text{G}} \qquad\qquad \frac{C \vee A \vee A}{C \vee A}\,\textsc{Fact}_{\text{G}}$$

where C and D are clauses and A is an atom.

The inference rules are schemas to describe all possible inferences of the inference system. $\textsc{Res}_{\text{G}}$ and $\textsc{Fact}_{\text{G}}$ are short names of the rules.

The idea of the resolution rule $\textsc{Res}_{\text{G}}$ is that we need to find a clause that contains an atom A positively (without $\neg$) and a clause that contains that same atom negatively (with $\neg$). Then we remove the A and the $\neg A$ from those clauses and collect all remaning literals from both clauses to form the conclusion.

The idea of the factoring rule $\textsc{Fact}_{\text{G}}$ is to take a clause that contains the same positive literal at least twice and remove one of the occurrences.

Exercise Convince yourself that the inference rules are sound, i.e., that the premises entail the conclusion.

Example

$$\begin{array}{ll}
\text{initial clauses} & \{\mathsf{p} \vee \mathsf{q}(\mathsf{a}),\ \neg\mathsf{p},\ \neg\mathsf{q}(\mathsf{a})\} \\
\dfrac{\mathsf{p} \vee \mathsf{q}(\mathsf{a}) \quad \neg\mathsf{q}(\mathsf{a})}{\mathsf{p}}\,\text{Res}_{\text{G}} & \{\mathsf{p} \vee \mathsf{q}(\mathsf{a}),\ \neg\mathsf{p},\ \neg\mathsf{q}(\mathsf{a}),\ \mathsf{p}\} \\
\dfrac{\mathsf{p} \quad \neg\mathsf{p}}{\bot}\,\text{Res}_{\text{G}} & \{\mathsf{p} \vee \mathsf{q}(\mathsf{a}),\ \neg\mathsf{p},\ \neg\mathsf{q}(\mathsf{a}),\ \mathsf{p},\ \bot\}
\end{array}$$

This is not the only way to derive the empty clause $\bot$ from the initial set of clauses: One could also start with the Res_{G} inference between $\mathsf{p} \vee \mathsf{q}(\mathsf{a})$ and $\neg\mathsf{p}$. This is bad because an automated prover must perform all possible inferences to make sure that it does not miss the essential ones. To keep our clause set small, we would like to allow just enough inferences to be able to derive the empty clause. We will see later how the *ordered* resolution calculus helps us remove some of these superfluous routes to the empty clause.

7.1.1 Proof Idea for Refutational Completeness

We will prove refutational completeness as follows. There are other proofs of refutational completeness of resolution, but this is the variant that can be extended to work for ordered resolution and resolution with redundancy as well.

- By Lemma 3, it suffices to show that any saturated set N of ground formulas with $\bot \notin N$ is satisfiable.
- We explicitly construct a model of a given N.
  - We start with the Herbrand interpretation corresponding to the empty set of ground atoms. (i.e., all predicates are always false)
  - We inspect each clause in N in a certain order and modify our interpretation so that the current clause is true, without changing the truth of previous clauses.
  - In the limit, we obtain a model of N.

7.1.2 Clause Order

- Choose an order $\succ$ of ground atoms that is total and well founded (no infinite descending chain).
- Extend the atom order to literals:
  - First compare the literal's atoms: $[\neg]A \succ [\neg]B$ if $A \succ B$
  - If the atoms are identical, the negative literal is larger: $\neg A \succ A$

- Extend the order to clauses:
  - To compare two clauses C and D, find the largest literal whose number of occurrences in C and D is different.
  - The clause that contains that literal more often is larger.

One can show that this clause order is a well-founded total order.

Example Let $\mathsf{q}(\mathsf{b}) \succ \mathsf{q}(\mathsf{a}) \succ \mathsf{p}$ be atoms. Then

$$\neg\mathsf{q}(\mathsf{b}) \succ \mathsf{q}(\mathsf{b}) \succ \neg\mathsf{q}(\mathsf{a}) \succ \mathsf{q}(\mathsf{a}) \succ \neg\mathsf{p} \succ \mathsf{p}$$

and

$$\mathsf{q}(\mathsf{a}) \vee \mathsf{q}(\mathsf{b}) \vee \mathsf{q}(\mathsf{b}) \;\succ\; \neg\mathsf{p} \vee \mathsf{q}(\mathsf{a}) \vee \mathsf{q}(\mathsf{b}) \;\succ\; \mathsf{p} \vee \mathsf{p} \vee \mathsf{q}(\mathsf{a}) \vee \mathsf{q}(\mathsf{b})$$

Definition 4. We call a literal L of a clause C *largest* if $L \succeq K$ for all literals K in C. We call a literal of a clause *strictly largest* if it is largest and occurs only once.

7.1.3 Refutational Completeness

Let $N \not\models \bot$ be a clause set and $\succ$ a clause order as defined above. We define sets I_C and Δ_C of ground atoms for all ground clauses C recursively w.r.t. $\succ$:

$$I_C := \bigcup_{D \prec C} \Delta_D$$

$$\Delta_C := \begin{cases} \{A\}, & \text{if } C \in N,\ I_C \not\models C, \text{ a positive literal } A \text{ is strictly largest in } C \\ \emptyset, & \text{otherwise} \end{cases}$$

If $\Delta_C = \{A\}$, we say that C is *productive* and *produces* A. Finally, let $I_N = \bigcup_C \Delta_C$ be our *candidate interpretation*.

Lemma 5.

(i) If $I_D \cup \Delta_D \models D$, then $I_C \models D$ for all $C \succ D$ and $I_N \models D$.

(ii) If $D = D' \vee A$ produces A, then $I_C \not\models D'$ for all $C \succcurlyeq D$.

Proof. (i) Let $I_D \cup \Delta_D \models D$. Then there must be a literal in D that is true in $I_D \cup \Delta_D$.

- If it is a positive literal A, then $A \in I_D \cup \Delta_D \subseteq I_C \subseteq I_N$ and thus $I_C \models D$ and $I_N \models D$.
- If it is a negative literal $\neg A$, then $A \notin I_D$. Since $\neg A \in D$, no clause $\succcurlyeq D$ can contain A as a strictly largest literal and thus no clause $\succcurlyeq D$ can produce A. Thus, $A \notin I_C$ and $A \notin I_N$ and therefore $I_C \models D$ and $I_N \models D$.

(ii) Let $D = D' \vee A$ produce A and let $C \succcurlyeq D$. By productivity, $I_D \not\models D'$ and thus $I_D \not\models L$ for every $L \in D'$. We must show that also $I_C \not\models L$ for every $L \in D'$.

- If $L = B$ is a positive literal, then $B \notin I_D$. Since A is strictly largest in $D = D' \vee A$ and $\succ$ is total, $B \prec A$. So no clause $\succcurlyeq D$ can contain B as a strictly largest literal and thus no clause $\succcurlyeq D$ can produce B. Thus, $B \notin I_C$ and therefore $I_C \not\models L$.
- If $L = \neg B$ is a negative literal, then $B \in I_D \subseteq I_C$ and thus $I_C \not\models L$. $\square$

Example Let $\mathsf{p} \prec \mathsf{q} \prec \mathsf{r}$, and $N = \{\neg\mathsf{p},\ \mathsf{p} \vee \mathsf{q},\ \neg\mathsf{q} \vee \mathsf{r} \vee \mathsf{r},\ \neg\mathsf{q} \vee \neg\mathsf{r}\}$. We order the clauses w.r.t. $\succ$, starting with the smallest clause, and construct the candidate model:

C	I_C	Δ_C	Remarks
$\neg\mathsf{p}$	$\emptyset$	$\emptyset$	$I_C \models C$
$\mathsf{p} \vee \mathsf{q}$	$\emptyset$	$\{\mathsf{q}\}$	$I_C \not\models C$, q strictly largest
$\neg\mathsf{q} \vee \mathsf{r} \vee \mathsf{r}$	$\{\mathsf{q}\}$	$\emptyset$	$I_C \not\models C$, r is largest, but not strictly!
$\neg\mathsf{q} \vee \neg\mathsf{r}$	$\{\mathsf{q}\}$	$\emptyset$	$I_C \models C$

However, the resulting interpretation $I_N = \{\mathsf{q}\}$ is not a model of $\neg\mathsf{q} \vee \mathsf{r} \vee \mathsf{r} \in N$. We will show that this can only happen when we forgot to perform an inference. And indeed, the conclusion of

$$\frac{\neg\mathsf{q} \vee \mathsf{r} \vee \mathsf{r}}{\neg\mathsf{q} \vee \mathsf{r}}\,\textsc{Fact}_{\textsc{G}}$$

is missing. Adding this clause to N we get the following candidate interpretation:

C	I_C	Δ_C	Remarks
$\neg\mathsf{p}$	$\emptyset$	$\emptyset$	$I_C \models C$
$\mathsf{p} \vee \mathsf{q}$	$\emptyset$	$\{\mathsf{q}\}$	$I_C \not\models C$, q strictly largest
$\neg\mathsf{q} \vee \mathsf{r}$	$\{\mathsf{q}\}$	$\{\mathsf{r}\}$	$I_C \not\models C$, r strictly largest
$\neg\mathsf{q} \vee \mathsf{r} \vee \mathsf{r}$	$\{\mathsf{q}, \mathsf{r}\}$	$\emptyset$	$I_C \models C$,
$\neg\mathsf{q} \vee \neg\mathsf{r}$	$\{\mathsf{q}, \mathsf{r}\}$	$\emptyset$	$I_C \not\models C$, but largest literal $\neg\mathsf{r}$ is negative!

This time, the resulting interpretation $I_N = \{\mathsf{q}, \mathsf{r}\}$ is not a model of $\neg\mathsf{q} \vee \neg\mathsf{r} \in N$. Again, this is because we forgot an inference:

$$\frac{\neg\mathsf{q} \vee \neg\mathsf{r} \quad \neg\mathsf{q} \vee \mathsf{r}}{\neg\mathsf{q}}\,\textsc{Res}_{\textsc{G}}$$

Lemma 6. *Let N be a ground clause set saturated w.r.t. ground resolution and $\bot \notin N$. Then for every $C \in N$, we have $I_C \cup \Delta_C \models C$.*

Proof. By well-founded induction on C w.r.t. $\succ$, we assume that for every $E \in N$ with $E \prec C$, we have $I_E \cup \Delta_E \models E$. Consider the largest literal of C:

Case 1: The largest literal A is positive and strictly largest. Then either $I_C \models C$ or $\Delta_C = \{A\}$. Either way, it follows that $I_C \cup \Delta_C \models C$.

Case 2: The largest literal A is positive, but not strictly largest—i.e., $C = C' \vee A \vee A$ for some C'. Since N is saturated, due to the inference

$$\frac{C' \vee A \vee A}{C' \vee A}\,\textsc{Fact}_{\text{G}}$$

the clause $E = C' \vee A$ must be in N. Since $E \prec C$, by our induction hypothesis, $I_E \cup \Delta_E \models E$. By Lemma 5(i), $I_C \models E$, and hence $I_C \models C$. Then C is not productive and $I_C \cup \Delta_C \models C$.

Case 3: The largest literal $\neg A$ is negative—i.e., $C = C' \vee \neg A$ for some C'. If $A \notin I_C$, then $I_C \models C$ and we are done. Otherwise, $A \in I_C$ and thus some clause $D = D' \vee A$ with $C \succ D$ must have produced A. Since N is saturated, due to the inference

$$\frac{D' \vee A \qquad C' \vee \neg A}{D' \vee C'}\,\textsc{Res}_{\text{G}}$$

the clause $E = D' \vee C'$ must be in N. We have $E \prec C$ (The largest literal around is $\neg A$ and $\neg A$ occurs less often in E than in C). By our induction hypothesis, $I_E \cup \Delta_E \models E$. By Lemma 5(i), $I_C \models E$, and by Lemma 5(ii), $I_C \not\models D'$. Thus, $I_C \models C'$ and hence $I_C \models C$. □

Theorem 7. *Ground resolution is refutationally complete.*

Proof. By Lemma 3, it suffices to show that for all all saturated $N \not\models \bot$, we have $I_N \models N$. By Lemma 6, for every $C \in N$, we have $I_C \cup \Delta_C \models C$. By Lemma 5(i), $I_N \models C$. □

7.1.4 Ordered ground resolution

Inspecting the proof of Lemma 6, we notice that we have used saturation of N only for certain kinds of $\textsc{Res}_{\text{G}}$ and $\textsc{Fact}_{\text{G}}$ inferences. This leads to ordered resolution, here in its ground version:

$$\frac{D \vee A \qquad C \vee \neg A}{D \vee C}\,\textsc{Res}_{\text{OG}}$$

where A is strictly largest in $D \vee A$ and $\neg A$ is largest in $C \vee \neg A$.

$$\frac{C \vee A \vee A}{C \vee A}\,\textsc{Fact}_{\text{OG}}$$

where A is largest in $C \vee A \vee A$.

This stricter variant of the calculus is still refutationally complete and the proof works as before.

7.2 (Nonground) Ordered Resolution

The real strength of resolution is that it can deal with variables. On ground formulas, SAT solvers are much more efficient.

Idea: a nonground clause C stands for all ground clauses that can be obtained by substituting its variables. These ground clauses are called *ground instances* $G(C)$. For example, if our signature contains a unary function f and two constants a and b, then the clause $\mathsf{p}(x) \vee \mathsf{q}(x)$ stands for:

$$\begin{array}{cc} \mathsf{p}(\mathsf{a}) \vee \mathsf{q}(\mathsf{a}) & \mathsf{p}(\mathsf{b}) \vee \mathsf{q}(\mathsf{b}) \\ \mathsf{p}(\mathsf{f}(\mathsf{a})) \vee \mathsf{q}(\mathsf{f}(\mathsf{a})) & \mathsf{p}(\mathsf{f}(\mathsf{b})) \vee \mathsf{q}(\mathsf{f}(\mathsf{b})) \\ \vdots & \vdots \end{array}$$

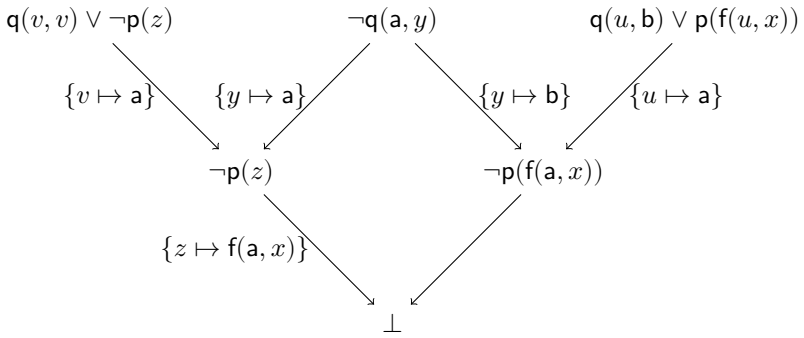
Based on this idea, we perform inferences on clauses with variables whenever there are corresponding inferences on their ground instances:

$$\frac{\mathsf{q} \vee \mathsf{p}(\mathsf{g}(x), y) \quad \mathsf{r}(z) \vee \neg \mathsf{p}(z, \mathsf{b})}{\mathsf{q} \vee \mathsf{r}(\mathsf{g}(x))}\,\textsc{Res}_{\mathrm{O}}$$

We perform this inference although the atoms $\mathsf{p}(\mathsf{g}(x), y)$ and $\mathsf{p}(z, \mathsf{b})$ are not the same because they can be made the same by substituting variables. There are infinitely many such substitutions, but the substitution $\{y \mapsto \mathsf{b},\ z \mapsto \mathsf{g}(x)\}$ captures all of them by leaving the substitution of x open. In general, for any two terms there is always such a substitution that captures all substitutions that would make the two terms equal, called the *most general unifier*.

In this way, we perform inferences on nonground clauses N, simulating inferences on their ground instances $G(N)$. Using this idea, we can show that $G(N) \vdash \bot$ implies $N \vdash \bot$. By refutational completeness of ground resolution, if $G(N) \models \bot$, then $G(N) \vdash \bot$. Moreover, one can show that $N \models \bot$ implies $G(N) \models \bot$. It follows that nonground resolution is also refutationally complete, i.e., that $N \models \bot$ implies $N \vdash \bot$.

Example Here is an example derivation for ordered resolution using three input clauses. We assume that atoms starting with q are larger than atoms starting with p. Each pair of arrows represents a $\textsc{Res}_{\mathrm{O}}$ inference.



8 Superposition

For first-order logic with equality, ordered resolution is quite inefficient. To reason about equality using ordered resolution, we would have to axiomatize the equality predicate: reflexivity, symmetry, transitivity, and congruence. With these axioms, lots of new inferences are possible, slowing down provers significantly.

The superposition calculus overcomes this issue with dedicated rules for equality. In fact, once we can deal with equality efficiently, we can assume that equality is the only predicate and encode other predicates $\mathsf{p}(t_1, \ldots, t_n)$ as $\mathsf{p}(t_1, \ldots, t_n) \approx \mathsf{true}$ using a function symbol p and a dummy symbol true. As a result, the literals of the clausal normal form for superposition are either equalities $s \approx t$ or disequalities $s \not\approx t$.

Instead of a literal order, superposition is parameterized by an order $\succ$ on terms that must fulfill certain properties. A literal order is induced by term order:

- Compare the largest term of one literal with the largest term of another literal
- If they are the same, compare the sign: negative literals are larger than positive ones.
- If the sign is the same, compare the remaining terms.

The ground version of superposition's inference rules are as follows:

$$\frac{D \vee t \approx t' \qquad C \vee [\neg]\, s[t] \approx s'}{D \vee C \vee [\neg]\, s[t'] \approx s'}\,\textsc{Sup}$$

where $t \succ t'$; $s[t] \succ s'$; and $t \approx t'$ and $[\neg]\, s[t] \approx s'$ are the largest literals in the premisses. Here, $s[t]$ stands for a term s that contains the subterm t. In the conclusion, $s[t']$ then represents the same term $s[t]$ with t replaced by t'. The notation $[\neg]$ means that the literal may be positive or negative, but the same sign must be used in premise and conclusion.

$$\frac{C \vee \neg\, s \approx s}{C}\,\textsc{EqRes}$$

where $\neg\, s \approx s$ is the largest literal in the premise.

$$\frac{C \vee s \approx t' \vee s \approx t}{C \vee \neg\, t \approx t' \vee s \approx t'}\,\textsc{EqFact}$$

where $s \succ t$ and $s \approx t$ is the largest literal in the premise.

Lifting of these rules to nonground clauses works in a similar fashion to resolution. However, Sup inferences are not necessary if the term t corresponds to a variable or is contained in a variable in the nonground clause.

Exercises

Exercise 1. Derive $\bot$ using the resolution calculus, starting with the following set of clauses:

1. $\neg\mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \neg\mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \mathsf{q}(\mathsf{b})$
2. $\mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \mathsf{q}(\mathsf{b})$
3. $\neg\mathsf{p}(\mathsf{g}(\mathsf{b},\mathsf{c})) \vee \neg\mathsf{q}(\mathsf{b})$
4. $\mathsf{p}(\mathsf{g}(\mathsf{b},\mathsf{c}))$

Exercise 2. Let a, b, c, d, e be predicates of arity 0. Consider the clauses

$$\neg\mathsf{a} \vee \neg\mathsf{b} \vee \neg\mathsf{c} \vee \neg\mathsf{d} \vee \neg\mathsf{e} \qquad \mathsf{a} \qquad \mathsf{b} \qquad \mathsf{c} \qquad \mathsf{d} \qquad \mathsf{e}$$

What is the number of clauses that can be derived by resolution from this set of clauses? What is the number of clauses that can be derived by *ordered* resolution with some arbitrary total order on the predicates?

Exercise 3. Prove the following statement using the superposition calculus:

> Assuming that $x^{-1} = 1/x$ for all $x \neq 0$ and that $\pi \neq 0$, we have $|1/\pi| = |\pi^{-1}|$.

In first-order logic, we can express the statement as follows:

$$\begin{aligned}&((\forall x.\, \neg x \approx \mathsf{zero} \rightarrow \mathsf{inv}(x) \approx \mathsf{div}(\mathsf{one}, x)) \;\wedge\; \neg\mathsf{pi} \approx \mathsf{zero})\\ &\quad\rightarrow\; \mathsf{abs}(\mathsf{div}(\mathsf{one}, \mathsf{pi})) \approx \mathsf{abs}(\mathsf{inv}(\mathsf{pi}))\end{aligned}$$

For the term order, simply order terms by the number of function symbols and variables they contain (counting multiple occurrences). (Hint: You will not need EqFact for this.)

Exercise 4. Let S be a function mapping each clause C to a subset of its negative literals. We call the literals in $S(C)$ the *selected* literals of C. We add the following restriction to ground (unordered) resolution: Res_G inferences are only performed if either $S(C \vee \neg A) = \emptyset$ or $\neg A \in S(C \vee \neg A)$. Fact_G inferences are only performed if $S(C \vee A \vee A) = \emptyset$. Show that Lemma 6 still holds using this revised inference system.

Solutions

Solution for Exercise 1

1.	$\neg \mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \neg \mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \mathsf{q}(\mathsf{b})$	(given)
2.	$\mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \mathsf{q}(\mathsf{b})$	(given)
3.	$\neg \mathsf{p}(\mathsf{g}(\mathsf{b},\mathsf{c})) \vee \neg \mathsf{q}(\mathsf{b})$	(given)
4.	$\mathsf{p}(\mathsf{g}(\mathsf{b},\mathsf{c}))$	(given)
5.	$\neg \mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \mathsf{q}(\mathsf{b}) \vee \mathsf{q}(\mathsf{b})$	($\textsc{Res}_{\mathrm{G}}$ 2. into 1.)
6.	$\neg \mathsf{p}(\mathsf{f}(\mathsf{c})) \vee \mathsf{q}(\mathsf{b})$	($\textsc{Fact}_{\mathrm{G}}$ 5.)
7.	$\mathsf{q}(\mathsf{b}) \vee \mathsf{q}(\mathsf{b})$	($\textsc{Res}_{\mathrm{G}}$ 2. into 6.)
8.	$\mathsf{q}(\mathsf{b})$	($\textsc{Fact}_{\mathrm{G}}$ 7.)
9.	$\neg \mathsf{p}(\mathsf{g}(\mathsf{b},\mathsf{c}))$	($\textsc{Res}_{\mathrm{G}}$ 8. into 3.)
10.	$\bot$	($\textsc{Res}_{\mathrm{G}}$ 4. into 9.)

Solution for Exercise 2 First, we consider unordered resolution. We can derive 5 clauses with 4 literals:

$$\neg \mathsf{b} \vee \neg \mathsf{c} \vee \neg \mathsf{d} \vee \neg \mathsf{e} \qquad \neg \mathsf{a} \vee \neg \mathsf{c} \vee \neg \mathsf{d} \vee \neg \mathsf{e}$$
$$\neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{d} \vee \neg \mathsf{e} \qquad \neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{c} \vee \neg \mathsf{e}$$
$$\neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{c} \vee \neg \mathsf{d}$$

From those, we can derive 10 clauses with 3 literals:

$$\neg \mathsf{c} \vee \neg \mathsf{d} \vee \neg \mathsf{e} \qquad \neg \mathsf{b} \vee \neg \mathsf{d} \vee \neg \mathsf{e} \qquad \neg \mathsf{b} \vee \neg \mathsf{c} \vee \neg \mathsf{e} \qquad \neg \mathsf{b} \vee \neg \mathsf{c} \vee \neg \mathsf{d}$$
$$\neg \mathsf{a} \vee \neg \mathsf{d} \vee \neg \mathsf{e} \qquad \neg \mathsf{a} \vee \neg \mathsf{c} \vee \neg \mathsf{e} \qquad \neg \mathsf{a} \vee \neg \mathsf{c} \vee \neg \mathsf{d} \qquad \neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{e}$$
$$\neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{d} \qquad \neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{c}$$

From those, we can derive 10 clauses with 2 literals:

$$\neg \mathsf{d} \vee \neg \mathsf{e} \qquad \neg \mathsf{c} \vee \neg \mathsf{e} \qquad \neg \mathsf{c} \vee \neg \mathsf{d} \qquad \neg \mathsf{b} \vee \neg \mathsf{e} \qquad \neg \mathsf{b} \vee \neg \mathsf{d}$$
$$\neg \mathsf{b} \vee \neg \mathsf{c} \qquad \neg \mathsf{a} \vee \neg \mathsf{e} \qquad \neg \mathsf{a} \vee \neg \mathsf{d} \qquad \neg \mathsf{a} \vee \neg \mathsf{c} \qquad \neg \mathsf{a} \vee \neg \mathsf{b}$$

From those, we can derive 5 clauses with 1 literal:

$$\neg \mathsf{a} \qquad \neg \mathsf{b} \qquad \neg \mathsf{c} \qquad \neg \mathsf{d} \qquad \neg \mathsf{e}$$

And finally the empty clause:

$$\bot$$

Including the 6 initial clauses, we have $6 + 5 + 10 + 10 + 5 + 1 = 37$ derivable clauses.

Next, we consider ordered resolution using the order $\mathsf{a} \prec \mathsf{b} \prec \mathsf{c} \prec \mathsf{d} \prec \mathsf{e}$. Now we are restricted to derive the following 5 clauses:

$$\neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{c} \vee \neg \mathsf{d} \qquad \neg \mathsf{a} \vee \neg \mathsf{b} \vee \neg \mathsf{c} \qquad \neg \mathsf{a} \vee \neg \mathsf{b} \qquad \neg \mathsf{a} \qquad \bot$$

Including the 6 initial clauses, we have $6 + 5 = 11$ derivable clauses.

Solution for Exercise 3 Negating and clausifying yields the following clauses:

$$x \approx \mathsf{zero} \lor \underline{\mathsf{div}(\mathsf{one}, x)} \approx \mathsf{inv}(x) \tag{1}$$

$$\underline{\mathsf{pi}} \not\approx \mathsf{zero} \tag{2}$$

$$\underline{\mathsf{abs}(\mathsf{div}(\mathsf{one}, \mathsf{pi}))} \neq \mathsf{abs}(\mathsf{inv}(\mathsf{pi})) \tag{3}$$

In each clause, underlining indicates the largest side of the maximal literals. In our examples, maximal simply means If we take x in clause (1) to be pi, the underlined term in (1) matches a subterm of the underlined term in clause (3). Thus, we can apply Sup using as premises the instance

$$\mathsf{pi} \approx \mathsf{zero} \lor \underline{\mathsf{div}(\mathsf{one}, \mathsf{pi})} \approx \mathsf{inv}(\mathsf{pi})$$

of clause (1) together with clause (3) to obtain

$$\mathsf{pi} \approx \mathsf{zero} \lor \underline{\mathsf{abs}(\mathsf{inv}(\mathsf{pi}))} \neq \underline{\mathsf{abs}(\mathsf{inv}(\mathsf{pi}))} \tag{4}$$

Next, we apply EqRes to clause (4) to generate

$$\underline{\mathsf{pi}} \approx \mathsf{zero} \tag{5}$$

Now that we have eliminated the larger literal of (4), we may work on the remaining smaller literal. There are multiple Sup inferences possible—e.g., between clauses (5) and (3) or between (5) and (4). The inference that leads to the desired contradiction, however, is a Sup inference between clauses (5) and (2) that produces

$$\underline{\mathsf{zero}} \neq \underline{\mathsf{zero}} \tag{6}$$

Finally, an EqRes inference on clause (6) yields the empty clause, which proves that the original lemma holds.

Solution for Exercise 4 If $S(C) = \emptyset$, we can proceed as in the original proof of Lemma 6. Otherwise, there exists some $\neg A \in S(C)$—i.e., $C = C' \lor \neg A$ for some C'. Note that $\neg A$ is not necessarily maximal. Nonetheless, we can proceed similarly to Case 3 of the original proof: If $A \notin I_C$, then $I_C \models C$ and we are done. Otherwise, $A \in I_C$ and thus some clause $D = D' \lor A$ with $C \succ D$ must have produced A. Since N is saturated, due to the inference

$$\frac{D' \lor A \qquad C' \lor \neg A}{D' \lor C'}\,\mathrm{Res_G}$$

the clause $E = D' \lor C'$ must be in N. This inference is permitted because $\neg A$ is selected in C. Since D produced A, A is strictly largest in D and thus all literals in D' are smaller than $\neg A$. It follows that the number of occurrences of $\neg A$ is smaller in E than in $C' \lor \neg A$ and therefore we have $E \prec C$. By our induction hypothesis, $I_E \cup \Delta_E \models E$. By Lemma 5(i), $I_C \models E$, and by Lemma 5(ii), $I_C \not\models D'$. Thus, $I_C \models C'$ and hence $I_C \models C$.