# First-Order Theorem Proving and Vampire

Laura Kovács

# Outline

Equality

Term Orderings

Non-Ground Case, Lifting

From Theory to Practice

Putting All Together, Summary

# First-order logic with equality

- Equality predicate: $=$.
- Equality: $l = r$.

The order of literals in equalities does not matter, that is, we consider an equality $l = r$ as a multiset consisting of two terms $l, r$, and so consider $l = r$ and $r = l$ equal.

# Equality. An Axiomatisation

- reflexivity axiom: $x = x$;
- symmetry axiom: $x = y \to y = x$;
- transitivity axiom: $x = y \wedge y = z \to x = z$;
- function substitution (congruence) axioms: $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \to f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$, for every function symbol $f$;
- predicate substitution (congruence) axioms: $x_1 = y_1 \wedge \ldots \wedge x_n = y_n \wedge P(x_1, \ldots, x_n) \to P(y_1, \ldots, y_n)$ for every predicate symbol $P$.

# Inference systems for logic with equality

We will next introduce a resolution and superposition inference system. This system is complete. One can eliminate redundancy.

# Inference systems for logic with equality

We will next introduce a resolution and superposition inference system. This system is complete. One can eliminate redundancy.

- Completeness is first established for ground clauses only.
- This can be “lifted” to arbitrary first-order clauses using a technique called lifting.
- Some notions (ordering, selection function) can first be defined for ground clauses and then “lift” them to non-ground clauses.

# Inference systems for logic with equality

We will next introduce a resolution and superposition inference system. This system is complete. One can eliminate redundancy.

- Completeness is first established for ground clauses only.
- This can be “lifted” to arbitrary first-order clauses using a technique called lifting.
- Some notions (ordering, selection function) can first be defined for ground clauses and then “lift” them to non-ground clauses.

# Simple Ground Superposition Inference System

Superposition: (right and left)

$$\frac{l = r \vee C \quad s[l] = t \vee D}{s[r] = t \vee C \vee D}\ (\text{Sup}), \quad \frac{l = r \vee C \quad s[l] \neq t \vee D}{s[r] \neq t \vee C \vee D}\ (\text{Sup}),$$

# Simple Ground Superposition Inference System

Superposition: (right and left)

$$\frac{l = r \vee C \quad s[l] = t \vee D}{s[r] = t \vee C \vee D}\ (\text{Sup}), \quad \frac{l = r \vee C \quad s[l] \neq t \vee D}{s[r] \neq t \vee C \vee D}\ (\text{Sup}),$$

Equality Resolution:

$$\frac{s \neq s \vee C}{C}\ (\text{ER}),$$

Equality Factoring:

$$\frac{s = t \vee s = t' \vee C}{s = t \vee t \neq t' \vee C}\ (\text{EF}),$$

# Example

$$f(a) = a \vee g(a) = a$$
$$f(f(a)) = a \vee g(g(a)) \neq a$$
$$f(f(a)) \neq a$$

# Example

$$f(a) = a \vee g(a) = a$$
$$f(f(a)) = a \vee g(g(a)) \neq a$$
$$f(f(a)) \neq a$$

# Can this system be used for efficient theorem proving?

Not really. It has too many inferences. For example, from the clause $f(a) = a$ we can derive any clause of the form

$$f^m(a) = f^n(a)$$

where $m, n \geq 0$.

# Can this system be used for efficient theorem proving?

Not really. It has too many inferences. For example, from the clause $f(a) = a$ we can derive any clause of the form

$$f^m(a) = f^n(a)$$

where $m, n \geq 0$.
Worst of all, the derived clauses can be much larger than the original clause $f(a) = a$.

# Can this system be used for efficient theorem proving?

Not really. It has too many inferences. For example, from the clause $f(a) = a$ we can derive any clause of the form

$$f^m(a) = f^n(a)$$

where $m, n \geq 0$.
Worst of all, the derived clauses can be much larger than the original clause $f(a) = a$.
The recipe is to use the previously introduced ingredients:

1. Ordering;
2. Literal selection;
3. Redundancy elimination.

# Atom and literal orderings on equalities

Equality atom comparison treats an equality $s = t$ as the multiset $\dot{\{}s, t\dot{\}}$.

- $(s' = t') \succ_{lit} (s = t)$ if $\dot{\{}s', t'\dot{\}} \succ \dot{\{}s, t\dot{\}}$
- $(s' \neq t') \succ_{lit} (s \neq t)$ if $\dot{\{}s', t'\dot{\}} \succ \dot{\{}s, t\dot{\}}$

with $\succ_{lit}$ being an induced ordering on literals (as in session 2).

# Ground Superposition Inference System $\mathbb{S}\mathrm{up}_{\succ,\sigma}$

Let $\sigma$ be a well-behaved literal selection function.
Superposition: (right and left)

$$\frac{l = r \vee C \quad s[l] = t \vee D}{s[r] = t \vee C \vee D}\ (\mathrm{Sup}), \qquad \frac{l = r \vee C \quad s[l] \neq t \vee D}{s[r] \neq t \vee C \vee D}\ (\mathrm{Sup}),$$

where (i) $l \succ r$, (ii) $s[l] \succ t$

# Ground Superposition Inference System $\mathbb{S}\mathrm{up}_{\succ,\sigma}$

Let $\sigma$ be a well-behaved literal selection function.
Superposition: (right and left)

$$\frac{l = r \vee C \quad s[l] = t \vee D}{s[r] = t \vee C \vee D}\ (\mathrm{Sup}), \qquad \frac{l = r \vee C \quad s[l] \neq t \vee D}{s[r] \neq t \vee C \vee D}\ (\mathrm{Sup}),$$

where (i) $l \succ r$, (ii) $s[l] \succ t$ , (iii) $l = r$ is strictly greater than any literal in $C$, (iv) (only for the superposition-right rule) $s[l] = t$ is greater than or equal to any literal in $D$.

# Ground Superposition Inference System $\mathbb{S}\mathrm{up}_{\succ,\sigma}$

Let $\sigma$ be a well-behaved literal selection function.

Superposition: (right and left)

$$\frac{l = r \vee C \quad s[l] = t \vee D}{s[r] = t \vee C \vee D}\ (\mathrm{Sup}), \qquad \frac{l = r \vee C \quad s[l] \neq t \vee D}{s[r] \neq t \vee C \vee D}\ (\mathrm{Sup}),$$

where (i) $l \succ r$, (ii) $s[l] \succ t$ , (iii) $l = r$ is strictly greater than any literal in $C$, (iv) (only for the superposition-right rule) $s[l] = t$ is greater than or equal to any literal in $D$.

Equality Resolution:

$$\frac{s \neq s \vee C}{C}\ (\mathrm{ER}),$$

Equality Factoring:

$$\frac{s = t \vee s = t' \vee C}{s = t \vee t \neq t' \vee C}\ (\mathrm{EF}),$$

where (i) $s \succ t \succeq t'$; (ii) $s = t$ is greater than or equal to any literal in $C$.

# Outline

Equality

Term Orderings

Non-Ground Case, Lifting

From Theory to Practice

Putting All Together, Summary

# Simplification Ordering

When we deal with equality, we need to work with term orderings.

# Simplification Ordering

When we deal with equality, we need to work with term orderings.

# Simplification Ordering

When we deal with equality, we need to work with term orderings.
Consider a strict ordering $\succ$ on signature symbols, such that $\succ$ is well-founded.
The ordering $\succ$ on terms is called a simplification ordering if

1. $\succ$ is well-founded;
2. $\succ$ is monotonic: if $l \succ r$, then $s[l] \succ s[r]$;
3. $\succ$ is stable under substitutions: if $l \succ r$, then $l\theta \succ r\theta$.

# A General Property of Term Orderings

If $\succ$ is a simplification ordering, then for every term $t[s]$ and its proper subterm $s$ we have $s \not\succ t[s]$.

# Knuth-Bendix Ordering (KBO), Ground Case

Let us fix

- Signature $\Sigma$, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on $\Sigma$, called precedence relation;
- Weight function $w : \Sigma \to \mathbb{N}$.

# Knuth-Bendix Ordering (KBO), Ground Case

Let us fix

- Signature $\Sigma$, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on $\Sigma$, called precedence relation;
- Weight function $w : \Sigma \to \mathbb{N}$.

Weight of a ground term $t$ is

$$|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|.$$

# Knuth-Bendix Ordering (KBO), Ground Case

Let us fix

- Signature $\Sigma$, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on $\Sigma$, called precedence relation;
- Weight function $w : \Sigma \to \mathbb{N}$.

Weight of a ground term $t$ is

$$|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|.$$

$g(t_1, \ldots, t_n) \succ_{KB} h(s_1, \ldots, s_m)$ if

# Knuth-Bendix Ordering (KBO), Ground Case

Let us fix

- Signature $\Sigma$, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on $\Sigma$, called precedence relation;
- Weight function $w : \Sigma \to \mathbb{N}$.

Weight of a ground term $t$ is

$$|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|.$$

$g(t_1, \ldots, t_n) \succ_{KB} h(s_1, \ldots, s_m)$ if

1. $|g(t_1, \ldots, t_n)| > |h(s_1, \ldots, s_m)|$ (by weight) or

# Knuth-Bendix Ordering (KBO), Ground Case

Let us fix

- Signature $\Sigma$, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on $\Sigma$, called precedence relation;
- Weight function $w : \Sigma \to \mathbb{N}$.

Weight of a ground term $t$ is

$$|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|.$$

$g(t_1, \ldots, t_n) \succ_{KB} h(s_1, \ldots, s_m)$ if

1. $|g(t_1, \ldots, t_n)| > |h(s_1, \ldots, s_m)|$ (by weight) or
2. $|g(t_1, \ldots, t_n)| = |h(s_1, \ldots, s_m)|$ and one of the following holds:
   - 2.1 $g \gg h$ (by precedence) or

# Knuth-Bendix Ordering (KBO), Ground Case

Let us fix

- Signature $\Sigma$, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on $\Sigma$, called precedence relation;
- Weight function $w : \Sigma \to \mathbb{N}$.

Weight of a ground term $t$ is

$$|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|.$$

$g(t_1, \ldots, t_n) \succ_{KB} h(s_1, \ldots, s_m)$ if

1. $|g(t_1, \ldots, t_n)| > |h(s_1, \ldots, s_m)|$ (by weight) or
2. $|g(t_1, \ldots, t_n)| = |h(s_1, \ldots, s_m)|$ and one of the following holds:
   - 2.1 $g \gg h$ (by precedence) or
   - 2.2 $g = h$ and for some $1 \leq i \leq n$ we have $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}$ and $t_i \succ_{KB} s_i$ (lexicographically).

# Example

$$\begin{aligned} w(a) &= 1 \\ w(b) &= 2 \\ w(f) &= 3 \\ w(g) &= 0 \end{aligned}$$

$$|f(g(a), f(a, b))|$$

# Example

$$\begin{aligned} w(a) &= 1 \\ w(b) &= 2 \\ w(f) &= 3 \\ w(g) &= 0 \end{aligned}$$

$$|f(g(a), f(a, b))| = |3(0(1), 3(1, 2))|$$

# Example

$$\begin{aligned} w(a) &= 1 \\ w(b) &= 2 \\ w(f) &= 3 \\ w(g) &= 0 \end{aligned}$$

$$|f(g(a), f(a, b))| = |3(0(1), 3(1, 2))| = 3 + 0 + 1 + 3 + 1 + 2$$

# Example

$$\begin{aligned} w(a) &= 1 \\ w(b) &= 2 \\ w(f) &= 3 \\ w(g) &= 0 \end{aligned}$$

$$|f(g(a), f(a, b))| = |3(0(1), 3(1, 2))| = 3 + 0 + 1 + 3 + 1 + 2 = 10.$$

# Example

$$\begin{aligned} w(a) &= 1 \\ w(b) &= 2 \\ w(f) &= 3 \\ w(g) &= 0 \end{aligned}$$

$$|f(g(a), f(a, b))| = |3(0(1), 3(1, 2))| = 3 + 0 + 1 + 3 + 1 + 2 = 10.$$

The Knuth-Bendix ordering is the main ordering used in Vampire and all other resolution and superposition theorem provers.

# Weight Functions, Ground Case

A weight function $w : \Sigma \to \mathbb{N}$ is any function satisfying:

- $w(a) > 0$ for any constant $a \in \Sigma$;
- if $w(f) = 0$ for a unary function $f \in \Sigma$, then $f \gg g$ for all functions $g \in \Sigma$ with $f \neq g$.
  That is, $f$ is the greatest element of $\Sigma$ wrt $\gg$.

As a consequence, there is at most one unary function $f$ with $w(f) = 0$.

# Another case of redundancy

Consider a superposition with a unit left premise:

$$\frac{\underline{l = r} \quad \underline{s[l] = t \vee D}}{s[r] = t \vee D}\ \text{(Sup)},$$

Note that we have

$$l = r, s[r] = t \vee D \models s[l] = t \vee D$$

# Another case of redundancy

Consider a superposition with a unit left premise:

$$\frac{\underline{l = r} \quad \underline{s[l] = t \vee D}}{s[r] = t \vee D} \text{ (Sup)},$$

Note that we have

$$l = r, s[r] = t \vee D \models s[l] = t \vee D$$

and we have

$$s[l] = t \vee D \succ s[r] = t \vee D.$$

# Another case of redundancy

Consider a superposition with a unit left premise:

$$\frac{\underline{l = r} \quad \underline{s[l] = t \vee D}}{s[r] = t \vee D} \text{ (Sup)},$$

Note that we have

$$l = r, s[r] = t \vee D \models s[l] = t \vee D$$

and we have

$$s[l] = t \vee D \succ s[r] = t \vee D.$$

If we also have $s[l] = t \vee D \succ l = r$, then the second premise is redundant and can be removed.

# Another case of redundancy

Consider a superposition with a unit left premise:

$$\frac{\underline{l = r} \quad \underline{s[l] = t \vee D}}{s[r] = t \vee D} \text{ (Sup)},$$

Note that we have

$$l = r, s[r] = t \vee D \models s[l] = t \vee D$$

and we have

$$s[l] = t \vee D \succ s[r] = t \vee D.$$

If we also have $s[l] = t \vee D \succ l = r$, then the second premise is redundant and can be removed.

This rule (superposition plus deletion) is sometimes called demodulation (also rewriting by unit equalities).

# Exercise – Ground Superposition-Based Theorem Proving

Consider the KBO ordering $\succ$ generated by:

– the precedence $f \gg a \gg b \gg c$;

and

– the weight function $w$ with $w(f) = w(a) = w(b) = w(c) = 1$.

Consider the set $S$ of ground formulas:

$$a = b \vee a = c$$
$$f(a) \neq f(b)$$
$$b = c$$

Apply saturation on S using an inference process based on the ground superposition calculus $\mathbb{S}\mathrm{up}_{\succ,\sigma}$ (including the inference rules of ground binary resolution with selection).

# Exercise – Ground Superposition-Based Theorem Proving

Consider the KBO ordering $\succ$ generated by:

– the precedence $f \gg a \gg b \gg c$;

and

– the weight function $w$ with $w(f) = w(a) = w(b) = w(c) = 1$.

Consider the set $S$ of ground formulas:

$$a = b \vee a = c$$
$$f(a) \neq f(b)$$
$$b = c$$

Apply saturation on S using an inference process based on the ground superposition calculus $\mathbb{S}\mathrm{up}_{\succ,\sigma}$ (including the inference rules of ground binary resolution with selection).

# Exercise – Ground Superposition-Based Theorem Proving

Consider the KBO ordering $\succ$ generated by:

– the precedence $f \gg a \gg b \gg c$;

and

– the weight function $w$ with $w(f) = w(a) = w(b) = w(c) = 1$.

Consider the set $S$ of ground formulas:

$$a = b \vee a = c$$
$$f(a) \neq f(b)$$
$$b = c$$

Apply saturation on S using an inference process based on the ground superposition calculus $\mathbb{S}\mathrm{up}_{\succ,\sigma}$ (including the inference rules of ground binary resolution with selection).

Challenge: Show that $S$ is unsatisfiable such that during saturation only 4 new clauses are generated.

# Outline

Equality

Term Orderings

Non-Ground Case, Lifting

From Theory to Practice

Putting All Together, Summary

# Lifting

Lifting is a technique for proving completeness theorems in the following way:

1. Prove completeness of the system for a set of ground clauses;
2. Lift the proof to the non-ground case.

# Lifting, Example

Consider two (non-ground) clauses $p(x, a) \vee q_1(x)$ and $\neg p(y, z) \vee q_2(y, z)$. If the signature contains function symbols, then both clauses have infinite sets of instances:

$$\begin{array}{rcl} \{p(r, a) \vee q_1(r) & \mid & r \text{ is ground}\} \\ \{\neg p(s, t) \vee q_2(s, t) & \mid & s, t \text{ are ground}\} \end{array}$$

We can resolve such instances if and only if $r = s$ and $t = a$. Then we can apply the following inference

$$\frac{p(s, a) \vee q_1(s) \quad \neg p(s, a) \vee q_2(s, a)}{q_1(s) \vee q_2(s, a)} \text{ (BR)}$$

But there is an infinite number of such inferences.

# Lifting, Idea

The idea is to represent an infinite number of ground inferences of the form

$$\frac{p(s,a) \vee q_1(s) \quad \neg p(s,a) \vee q_2(s,a)}{q_1(s) \vee q_2(s,a)} \text{ (BR)}$$

by a single non-ground inference

$$\frac{p(x,a) \vee q_1(x) \quad \neg p(y,z) \vee q_2(y,z)}{q_1(y) \vee q_2(y,a)} \text{ (BR)}$$

Is this always possible?

# Yes!

$$\frac{p(x,a) \vee q_1(x) \quad \neg p(y,z) \vee q_2(y,z)}{q_1(y) \vee q_2(y,a)} \text{ (BR)}$$

Note that the substitution $\{x \mapsto y, z \mapsto a\}$ is a solution of the "equation" $p(x,a) = p(y,z)$.

# Lifting (Robinson, 1965)

## Lifting Lemma for BR in $\mathbb{BR}$:

Let $C$ and $D$ clauses with no shared variables. If:

$$\dfrac{\begin{array}{cc} C & D \\ \downarrow\sigma_1 & \downarrow\sigma_2 \\ C\sigma_1 & D\sigma_2 \end{array}}{C'} \text{ (\textit{ground} BR)}$$

# Lifting (Robinson, 1965)

## Lifting Lemma for BR in $\mathbb{BR}$:

Let $C$ and $D$ clauses with no shared variables. If:

$$\dfrac{\begin{array}{cc} C & D \\ \downarrow\sigma_1 & \downarrow\sigma_2 \\ C\sigma_1 & D\sigma_2 \end{array}}{C'} \ (\textit{ground}\ \text{BR})$$

then there exists an substitution $\sigma$ sucht that:

$$\dfrac{C \quad D}{C''} \ (\textit{non}-\textit{ground}\ \text{BR})$$

$$\downarrow\sigma$$

$$C' = C''\sigma$$

$\sigma$ is a most general unifier (mgu) of $C, D$.

# Lifting (Robinson, 1965)(Bachmair & Ganzinger, 1990)

## Lifting Lemma for BR in $\mathbb{BR}$:

Let $C$ and $D$ clauses with no shared variables. If:

$$\frac{\begin{array}{cc} C & D \\ \downarrow\sigma_1 & \downarrow\sigma_2 \\ C\sigma_1 & D\sigma_2 \end{array}}{C'} \ (\textit{ground}\ \text{BR})$$

then there exists an substitution $\sigma$ sucht that:

$$\frac{C \quad D}{C''} \ (\textit{non}-\textit{ground}\ \text{BR})$$

$$\downarrow\sigma$$

$$C' = C''\sigma$$

Lifting: ground inferences of $\mathbb{BR}_{\succ,\sigma}$, $\mathbb{Sup}_{\succ,\sigma}$,
orderings $\succ$, selection functions $\sigma$

# Non-Ground Superposition, Lifting

Superposition:

$$\frac{l = r \vee C \quad s[l'] = t \vee D}{(s[r] = t \vee C \vee D)\theta}\ (\text{Sup}), \qquad \frac{l = r \vee C \quad s[l'] \neq t \vee D}{(s[r] \neq t \vee C \vee D)\theta}\ (\text{Sup}),$$

where

1. $\theta$ is an mgu of $l$ and $l'$;
2. $l'$ is not a variable;
3. $r\theta \not\succeq l\theta$;
4. $t\theta \not\succeq s[l']\theta$.

# Non-Ground Superposition, Lifting

Superposition:

$$\frac{l = r \vee C \quad s[l'] = t \vee D}{(s[r] = t \vee C \vee D)\theta}\ (\text{Sup}), \qquad \frac{l = r \vee C \quad s[l'] \neq t \vee D}{(s[r] \neq t \vee C \vee D)\theta}\ (\text{Sup}),$$

where

1. $\theta$ is an mgu of $l$ and $l'$;
2. $l'$ is not a variable;
3. $r\theta \not\succeq l\theta$;
4. $t\theta \not\succeq s[l']\theta$.

Observations:

- ordering is partial, hence conditions like $r\theta \not\succeq l\theta$;
- these conditions must be checked a posteriori, that is, after the rule has been applied.

Note, however, that $l \succ r$ implies $l\theta \succ r\theta$, so checking orderings a priory helps.

# More rules

Equality Resolution:

$$\frac{s \neq s' \vee C}{C\theta} \text{ (ER)},$$

where $\theta$ is an mgu of $s$ and $s'$.

Equality Factoring:

$$\frac{l = r \vee l' = r' \vee C}{(l = r \vee r \neq r' \vee C)\theta} \text{ (EF)},$$

where $\theta$ is an mgu of $l$ and $l'$, $r\theta \not\succeq l\theta$, $r'\theta \not\succeq l\theta$, and $r'\theta \not\succeq r\theta$.

# Outline

Equality

Term Orderings

Non-Ground Case, Lifting

From Theory to Practice

Putting All Together, Summary

# From theory to practice

- Preprocessing and CNF transformation;
- Superposition system;
- Orderings;
- Selection functions;
- Fairness (saturation algorithms);
- Redundancy.

# Vampire's preprocessing (incomplete list)

1. (Optional) Select a relevant subset of formulas.
2. (Optional) Add theory axioms;
3. Rectify the formula.
4. If the formula contains any occurrence of $\top$ or $\bot$, simplify the formula.
5. Remove if-then-else and let-in connectives.
6. Apply pure predicate elimination.
7. (Optional) Remove unused predicate definitions.
8. Convert the formula into equivalence negation normal form (ENNF).
9. Use a naming technique to replace some subformulas by their names.
10. Convert the formula into negation normal form (NNF).
11. Skolemize the formula.
12. (Optional) Replace equality axioms.
13. Determine a literal ordering to be used.
14. Transform the formula into its clausal normal form.
15. Remove tautologies.
16. Pure literal elimination.

# How to Design a Good Saturation Algorithm?

A saturation algorithm must be fair: every possible generating inference must eventually be selected.

Two main implementation principles:

apply simplifying inferences eagerly;
apply generating inferences lazily.

checking for simplifying inferences should pay off;
so it must be cheap.

# How to Design a Good Saturation Algorithm?

A saturation algorithm must be fair: every possible generating inference must eventually be selected.

Two main implementation principles:

apply simplifying inferences eagerly;
apply generating inferences lazily.

checking for simplifying inferences should pay off;
so it must be cheap.

Try: `./vampire -fsr off -awr 4:1 GRP140-1.p`

# Outline

Equality

Term Orderings

Non-Ground Case, Lifting

From Theory to Practice

Putting All Together, Summary

Revisit the group theory example of session 1

```
./vampire -stat full group.tptp
```