Outline

Equality

First-order logic with equality

- Equality predicate: $\simeq$.
- Equality: $l \simeq r$.

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

Equality. An Axiomatisation

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

Inference systems for logic with equality

We will define a resolution and superposition inference system. This system is complete. One can eliminate redundancy (but the literal ordering needs to satisfy additional properties).

Inference systems for logic with equality

We will define a resolution and superposition inference system. This system is complete. One can eliminate redundancy (but the literal ordering needs to satisfy additional properties).
Moreover, we will first define it only for ground clauses. On the theoretical side,

- Completeness is first proved for ground clauses only.
- It is then “lifted” to arbitrary clauses using a technique called lifting.
- Moreover, this way some notions (ordering, selection function) can first be defined for ground clauses only and then it is relatively easy to see how to generalise them for non-ground clauses.

Simple Ground Superposition Inference System

Superposition: (right and left)

$$\frac{l \simeq r \vee C \quad s[l] \simeq t \vee D}{s[r] \simeq t \vee C \vee D} \text{ (Sup)}, \quad \frac{l \simeq r \vee C \quad s[l] \not\simeq t \vee D}{s[r] \not\simeq t \vee C \vee D} \text{ (Sup)},$$

Simple Ground Superposition Inference System

Superposition: (right and left)

$$\frac{l \simeq r \vee C \quad s[l] \simeq t \vee D}{s[r] \simeq t \vee C \vee D} \text{ (Sup)}, \quad \frac{l \simeq r \vee C \quad s[l] \not\simeq t \vee D}{s[r] \not\simeq t \vee C \vee D} \text{ (Sup)},$$

Equality Resolution:

$$\frac{s \not\simeq s \vee C}{C} \text{ (ER)},$$

Simple Ground Superposition Inference System

Superposition: (right and left)

$$\frac{l \simeq r \vee C \quad s[l] \simeq t \vee D}{s[r] \simeq t \vee C \vee D} \text{ (Sup)}, \quad \frac{l \simeq r \vee C \quad s[l] \not\simeq t \vee D}{s[r] \not\simeq t \vee C \vee D} \text{ (Sup)},$$

Equality Resolution:

$$\frac{s \not\simeq s \vee C}{C} \text{ (ER)},$$

Equality Factoring:

$$\frac{s \simeq t \vee s \simeq t' \vee C}{s \simeq t \vee t \not\simeq t' \vee C} \text{ (EF)},$$

Example

$$f(a) \simeq a \vee g(a) \simeq a$$
$$f(f(a)) \simeq a \vee g(g(a)) \not\simeq a$$
$$f(f(a)) \not\simeq a$$

Can this system be used for efficient theorem proving?

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

$$f^m(a) \simeq f^n(a)$$

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 $f(a) \simeq a$ we can derive any clause of the form

$$f^m(a) \simeq f^n(a)$$

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

Can this system be used for efficient theorem proving?

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

$$f^m(a) \simeq f^n(a)$$

where $m, n \geq 0$.
Worst of all, the derived clauses can be much larger than the original clause $f(a) \simeq a$.
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 $s \simeq t$ as the multiset $\dot{\{}s, t\dot{\}}$.

- $(s' \simeq t') \succ_{lit} (s \simeq t)$ if $\dot{\{}s', t'\dot{\}} \succ \dot{\{}s, t\dot{\}}$.
- $(s' \not\simeq t') \succ_{lit} (s \not\simeq t)$ if $\dot{\{}s', t'\dot{\}} \succ \dot{\{}s, t\dot{\}}$.

Finally, we assert that all non-equality literals be greater than all equality literals.

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

Let σ be a literal selection function.
Superposition: (right and left)

$$\frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \simeq t} \vee D}{s[r] \simeq t \vee C \vee D}\ (\text{Sup}), \quad \frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \not\simeq t} \vee D}{s[r] \not\simeq t \vee C \vee D}\ (\text{Sup}),$$

where (i) $l \succ r$, (ii) $s[l] \succ t$, (iii) $l \simeq r$ is strictly greater than any literal in C, (iv) $s[l] \simeq t$ is greater than or equal to any literal in D.

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

Let σ be a literal selection function.

Superposition: (right and left)

$$\frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \simeq t} \vee D}{s[r] \simeq t \vee C \vee D}\ \text{(Sup)}, \quad \frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \not\simeq t} \vee D}{s[r] \not\simeq t \vee C \vee D}\ \text{(Sup)},$$

where (i) $l \succ r$, (ii) $s[l] \succ t$, (iii) $l \simeq r$ is strictly greater than any literal in C, (iv) $s[l] \simeq t$ is greater than or equal to any literal in D.

Equality Resolution:

$$\frac{\underline{s \not\simeq s} \vee C}{C}\ \text{(ER)},$$

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

Let σ be a literal selection function.
Superposition: (right and left)

$$\frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \simeq t} \vee D}{s[r] \simeq t \vee C \vee D} \text{ (Sup)}, \quad \frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \not\simeq t} \vee D}{s[r] \not\simeq t \vee C \vee D} \text{ (Sup)},$$

where (i) $l \succ r$, (ii) $s[l] \succ t$, (iii) $l \simeq r$ is strictly greater than any literal in C, (iv) $s[l] \simeq t$ is greater than or equal to any literal in D.
Equality Resolution:

$$\frac{\underline{s \not\simeq s} \vee C}{C} \text{ (ER)},$$

Equality Factoring:

$$\frac{\underline{s \simeq t} \vee s \simeq t' \vee C}{s \simeq t \vee t \not\simeq t' \vee C} \text{ (EF)},$$

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

Extension to arbitrary (non-equality) literals

- Consider a two-sorted logic in which equality is the only predicate symbol.
- Interpret terms as terms of the first sort and non-equality atoms as terms of the second sort.
- Add a constant $\top$ of the second sort.
- Replace non-equality atoms $p(t_1, \ldots, t_n)$ by equalities of the second sort $p(t_1, \ldots, t_n) \simeq \top$.

Extension to arbitrary (non-equality) literals

- Consider a two-sorted logic in which equality is the only predicate symbol.
- Interpret terms as terms of the first sort and non-equality atoms as terms of the second sort.
- Add a constant $\top$ of the second sort.
- Replace non-equality atoms $p(t_1, \ldots, t_n)$ by equalities of the second sort $p(t_1, \ldots, t_n) \simeq \top$.

For example, the clause

$$p(a, b) \vee \neg q(a) \vee a \neq b$$

becomes

$$p(a, b) \simeq \top \vee q(a) \not\simeq \top \vee a \neq b.$$

Binary resolution inferences can be represented by inferences in the superposition system

We ignore selection functions.

$$\frac{A \vee C_1 \quad \neg A \vee C_2}{C_1 \vee C_2} \text{ (BR)}$$

$$\cfrac{\cfrac{A \simeq \top \vee C_1 \quad A \not\simeq \top \vee C_2}{\top \not\simeq \top \vee C_1 \vee C_2} \text{ (Sup)}}{C_1 \vee C_2} \text{ (ER)}$$

Exercise

Positive factoring can also be represented by inferences in the superposition system.

Simplification Ordering

The only restriction we imposed on term orderings was well-foundedness and stability under substitutions. When we deal with equality, these two properties are insufficient. We need a third property, called monotonicity.

An 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 $l\theta \succ r\theta$.

Simplification Ordering

The only restriction we imposed on term orderings was well-foundedness and stability under substitutions. When we deal with equality, these two properties are insufficient. We need a third property, called monotonicity.
An 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 $l\theta \succ r\theta$.

One can combine the last two properties into one:

2a. If $l \succ r$, then $s[l\theta] \succ s[r\theta]$.

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 $s \not\succ t[s]$.

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 $s \not\succ t[s]$.

Consider an example.

$$
\begin{array}{l}
f(a) \simeq a \\
f(f(a)) \simeq a \\
f(f(f(a))) \simeq a
\end{array}
$$

Then both $f(f(a)) \simeq a$ and $f(f(f(a))) \simeq a$ are redundant. The clause $f(a) \simeq a$ is a logical consequence of $\{f(f(a)) \simeq a, f(f(f(a))) \simeq a\}$ but is not redundant.

Term Algebra

Term algebra $TA(\Sigma)$ of signature Σ:

- Domain: the set of all ground terms of Σ.
- Interpretation of any function symbol f or constant c is defined as follows::

$$\begin{aligned} f_{TA(\Sigma)}(t_1, \ldots, t_n) &\stackrel{\text{def}}{\Leftrightarrow} f(t_1, \ldots, t_n); \\ c_{TA(\Sigma)} &\stackrel{\text{def}}{\Leftrightarrow} c. \end{aligned}$$

Knuth-Bendix Ordering, Ground Case

Let us fix

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

Knuth-Bendix Ordering, Ground Case

Let us fix

- Signature Σ, it induces the term algebra $TA(\Sigma)$.
- Total ordering $\gg$ on Σ, 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, Ground Case

Let us fix

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

Weight of a ground term t is

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

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

Knuth-Bendix Ordering, Ground Case

Let us fix

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

Weight of a ground term t is

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

$g(t_1, \dots, t_n) \succ_{KB} h(s_1, \dots, s_n)$ if

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

Knuth-Bendix Ordering, Ground Case

Let us fix

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

Weight of a ground term t is

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

$g(t_1, \dots, t_n) \succ_{KB} h(s_1, \dots, s_n)$ if

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

Knuth-Bendix Ordering, Ground Case

Let us fix

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

Weight of a ground term t is

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

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

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

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

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

There exists also a non-ground version of the Knuth-Bendix ordering and a (nearly) linear time algorithm for term comparison using this ordering.

Example

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

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

There exists also a non-ground version of the Knuth-Bendix ordering and a (nearly) linear time algorithm for term comparison using this ordering.
The Knuth-Bendix ordering is the main ordering used on Vampire and all other resolution and superposition theorem provers.

Same Property

The conclusion is strictly smaller than the rightmost premise:

$$\frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \simeq t} \vee D}{s[r] \simeq t \vee C \vee D}\ (\mathrm{Sup}), \quad \frac{\underline{l \simeq r} \vee C \quad \underline{s[l] \not\simeq t} \vee D}{s[r] \not\simeq t \vee C \vee D}\ (\mathrm{Sup}),$$

where (i) $l \succ r$, (ii) $s[l] \succ t$, (iii) $l \simeq r$ is strictly greater than any literal in C, (iv) $s[l] \simeq t$ is greater than or equal to any literal in D.

New redundancy

Consider a superposition with a unit left premise:

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

Note that we have

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

New redundancy

Consider a superposition with a unit left premise:

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

Note that we have

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

and we have

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

New redundancy

Consider a superposition with a unit left premise:

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

Note that we have

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

and we have

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

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

New redundancy

Consider a superposition with a unit left premise:

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

Note that we have

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

and we have

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

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

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