Geometric Resolution: A Proof Procedure Based on Finite Model Search

Hans de Nivelle

Max Planck Institut für Informatik, Saarbrücken, Germany,

Canberra, 30.11.2006

(Joint work with Jia Meng, from here)

Introduction

We present a new calculus for theorem proving in first-order logic with equality.

We call the calculus geometric resolution, because it operates on a normal form, which is derived from geometric formulas. (this is a first-order fragment introduced by Thoralf Skolem)

We show that the calculus is sound and complete for first-order logic.

Disclaimer

Geometric resolution has nothing to do with computer graphics!

Motivation

- Try out something new.
- Avoid use of Herbrand's theorem, because (unrestricted) interpretations can be much more compact than Herbrand interpretations.
- Find general theorem proving strategies with good termination behaviour, and which give more information in the case of termination.
- Find theorem proving strategies that can deal better with partial functions, and incompletely defined functions.

Geometric Formulas

Definition: We assume an infinite set of variables $\mathcal{V}$.

A variable atom is an atom of one of the following two forms:

1. $p(v_1, \ldots, v_n)$ with $n \geq 0$ and $v_1, \ldots, v_n \in \mathcal{V}$.
2. $v_1 \not\approx v_2$ with $v_1, v_2 \in \mathcal{V}$.

Observe that:

- There are no positive equalities.
- There are no constants and no function symbols.

Definition: A geometric formula has form

$$\forall \overline{x}\ A_1(\overline{x}) \wedge \cdots \wedge A_p(\overline{x}) \wedge x_1 \not\approx x_1' \wedge \cdots \wedge x_q \not\approx x_q' \rightarrow Z(\overline{x}),$$

in which $x_1, x_1', \ldots, x_q, x_q' \in \overline{x} \subseteq \mathcal{V}$.

$Z(\overline{x})$ can have one of the following three forms:

1. The false constant $\perp$.
2. A disjunction of atoms $B_1(\overline{x}) \vee \cdots \vee B_r(\overline{x})$, with $r > 0$.
3. An existential formula of form $\exists y\ B(\overline{x}, y)$.

Types 1 and 2 overlap (if one would allow $r = 0$) but we prefer to distinguish the types. Geometric formulas of Type 1 are called lemmas. Formulas of Type 2 are called disjunctive. Formulas of Type 3 are called existential.

Example 1

We might be interested in finding out whether
$a \approx b,\ b \approx c \ \vdash\ a \approx c.$

We try to find a model for

$$a \approx b, \quad b \approx c, \quad a \not\approx c.$$

Resulting geometric formulas are:

$A(X) \wedge B(Y) \wedge X \not\approx Y \rightarrow \bot,$
$B(X) \wedge C(Y) \wedge X \not\approx Y \rightarrow \bot,$
$A(X) \wedge C(X) \rightarrow \bot,$

$\rightarrow \exists x\ A(x),$
$\rightarrow \exists x\ B(x),$
$\rightarrow \exists x\ C(x).$

Example 2

What about $s(a) \approx a \vdash s(s(a)) \approx a$?

Try to find model for

$$s(a) \approx a, \quad s(s(a)) \not\approx a.$$

$A(X) \land S(X, Y) \land A(Y) \land X \not\approx Y \to \bot,$
$A(X) \land S(X, Y) \land S(Y, X) \to \bot,$

$\exists x \; A(x),$
$\forall x \exists y \; S(x, y).$

Example 3

$a \approx s(a), \quad p(a,a) \vee p(s(a), s(a)) \vdash p(a,a).$

Negation of goal:

$$a \approx s(a), \quad p(a,a) \vee p(s(a), s(a)), \quad \neg p(a,a).$$

$A(X) \wedge S(X,Y) \wedge X \not\approx Y \rightarrow \bot,$
$A(X) \wedge S(X,Y) \rightarrow p(X,X) \vee p(Y,Y),$
$A(X) \wedge p(X,X) \rightarrow \bot,$

$\exists x\ A(x),$
$\forall x \exists y\ S(x,y).$

After these examples, you might be willing to believe that:

Theorem:

Every set of first-order formulas can be translated into a set of geometric formulas, which is equisatisfiable.

The result (and the computation) can be linear in the size of the input.

- First compute negation normal form,
- Reduce scope if quantifiers (if possible).
- Then follows the most interesting step of the transformation, which is anti-Skolemization:

- For each function symbol or constant f, introduce a new predicate symbol P_f, s.t. $\#P_f = \#f + 1$.
- for each new predicate symbol P_f, introduce a seriality axiom:

$$\forall \overline{x} \; \exists y \; P_f(\overline{x}, y).$$

- As long as F contains a functional term, let $f(x_1, \ldots, x_n)$ be a functional term with only variable arguments.

  Write $F = F[\ A[\ f(x_1, \ldots, x_n)\ ]\ ]$, where A is the smallest subformula that contains all occurrences of $f(x_1, \ldots, x_n)$.

  Replace

$$F[\ A[\ f(x_1, \ldots, x_n)\ ]\ ]$$

  by

$$F[\ \forall y\ (P_f(x_1, \ldots, x_n, y) \rightarrow A[y]\ )\ ].$$

Example (of anti-Skolemization)

The formula

$$\forall x \exists y \; y \approx s(s(x))$$

is replaced by

$$\forall x \alpha \;\; S(x, \alpha) \to \exists y \; y \approx s(\alpha).$$

One more replacement results in

$$\forall x \alpha \beta \;\; S(x, \alpha) \wedge S(\alpha, \beta) \to \exists y \; y \approx \beta.$$

The seriality axiom is:

$$\forall x \exists y \; S(x, y).$$

Another Example

$$\forall x \exists y \;\; t(x,y) \approx n,$$

$\Rightarrow$

$$\forall x \exists y \forall \alpha \;\; T(x,y,\alpha) \rightarrow \alpha \approx n,$$

$\Rightarrow$

$$\forall \beta \; N(\beta) \rightarrow \forall x \exists y \forall \alpha \;\; T(x,y,\alpha) \rightarrow \alpha \approx \beta.$$

The seriality axioms are:

$$\forall xy \exists z \; T(x,y,z),$$

$$\exists x \; N(x).$$

Searching for a Model

Definition: An interpretation is a finite set of atoms, with arguments from a fixed, given set $\mathcal{E}$.

Equality is interpreted as object equality, therefore there are no disequality atoms in interpretations.

Examples of interpretations are

$$A(e_0),\quad S(e_0, e_1),\quad S(e_1, e_2),\quad B(e_2).$$

$$A(e_0),\quad B(e_1),\quad P(e_0, e_1, e_2),\quad Q(e_2, e_2, e_1).$$

A Naive Algorithm for Theorem Proving

Definition: Let I be an interpretation. We call geometric formula

$$\forall \overline{x}\ A_1(\overline{x}) \wedge \cdots \wedge A_p(\overline{x}) \wedge x_1 \not\approx x_1' \wedge \cdots \wedge x_q \not\approx x_q' \rightarrow Z(\overline{x})$$

applicable in I with ground substitution Θ, if

- All $A_i(\overline{x})\Theta$ are in I.
- For each $x_j \not\approx x_j'$, $\quad x_j\Theta$ and $x_j'\Theta$ are distinct.
- and $Z(\overline{x})\Theta$ is false in I. (definition follows on next slide)

When is $Z(\overline{x})\Theta$ false in I?

1. If $Z(\overline{x})$ has form $\bot$, then $Z(\overline{x})\Theta$ is always false in I.
2. If $Z(\overline{x})$ has form $B_1(\overline{x}) \vee \cdots \vee B_r(\overline{x})$ then $Z(\overline{x})\Theta$ is false in I, if none of $B_j(\overline{x})\Theta$ is present in I.
3. If $Z(\overline{x})$ has form $\exists y\ B(\overline{x}, y)$ then $Z(\overline{x})\Theta$ is false in I if there is no element $e \in \mathcal{E}$, s.t. $(B(\overline{x}, y)\Theta)\ \{y := e\}$ is present in I.

Start with empty interpretation $I = \{ \}$.

- If there is no applicable rule, then I is a model.
- Otherwise, select a rule $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow Z(\overline{x})$ that is applicable on I with ground substitution Θ.
  - If $Z(\overline{x})$ has form $\perp$, then backtrack.
  - If $Z(\overline{x})$ has form $B_1(\overline{x}) \vee \cdots \vee B_r(\overline{x})$, then backtrack through all of
    $$I \cup \{B_j(\overline{x})\Theta\}.$$
  - If $Z(\overline{x})$ has form $\exists y\ B(\overline{x}, y)$, then backtrack through
    $$I \cup \{\ \ B(\overline{x}, y)\ \Theta \cdot \{x := e\}\ \ \},$$
    for each e that is present in I. In addition, try
    $$I \cup \{\ \ B(\overline{x}, y)\ \Theta \cdot \{x := \hat{e}\}\ \ \}$$
    for a new element $\hat{e}$ that is not present in I.

Remember the example

$A(X) \wedge B(Y) \wedge X \not\approx Y \to \bot, \quad B(X) \wedge C(Y) \wedge X \not\approx Y \to \bot,$
$A(X) \wedge C(X) \to \bot,$

$\to \exists x\ A(x), \quad \to \exists x\ B(x), \quad \to \exists x\ C(x).$

(empty interpretation),
$A(e_0)$,
$A(e_0),\ B(e_0)$,
$A(e_0),\ B(e_0),\ C(e_0)$,
$A(e_0),\ B(e_0),\ C(e_1)$,
$A(e_0),\ B(e_1)$.

(backtracking complete)

An example with disjunction:

$\rightarrow \exists x\ A(x),$
$A(X) \rightarrow B(X) \vee C(X), \quad A(X) \wedge B(X) \rightarrow \perp, \quad C(X) \rightarrow \perp.$

(empty interpretation),
$A(e_0),$
$A(e_0),\ B(e_0),$
$A(e_0),\ C(e_0).$

(backtracking complete)

Evaluation of the Naive Model Search Algorithm

- A clever implementation of naive model search performs better than I expected.
- Much depends on the selection strategy. (i.e. which applicable rule is expanded first)
- But, of course, this algorithm will never be seriously competitive.

How to improve?

$\Rightarrow$ Avoid work being redone, add learning.

Model Search with Learning

The naive search algorithm attempts to construct an interpretation I using backtracking. It maintains a set of geometric formulas $\mathcal{G}$ and an interpretation I, which it tries to extend to a model.

Let us call a recursive implementation **search**$(I, \mathcal{G})$.

The improved version **search**$^{+}(I, \mathcal{G})$ has the following specification:

At every time when it returns (including returns from recursive calls) :

Either I has been extended to a complete model (no rules in $\mathcal{G}$ are applicable),
or $\mathcal{G}$ has been extended in such a way that there is a rule of form $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \bot$ in $\mathcal{G}$, which is applicable in I.

The algorithm $\mathbf{search}^+(I, \mathcal{G})$ is implemented as follows:

- If I is a model, then return I.
- Otherwise, there exists a rule $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow Z(\overline{x})$ that is applicable on I with ground substitution Θ.
- If $Z(\overline{x}) = \perp$, then we return the rule as is.
- If $Z(\overline{x})$ has form $B_1(\overline{x}) \vee \cdots \vee B_r(\overline{x})$, then recursively call

$$\mathbf{search}^+(I \cup \{B_1(\overline{x}\Theta)\}, \mathcal{G}), \ldots, \mathbf{search}^+(I \cup \{B_r(\overline{x}\Theta)\}, \mathcal{G}).$$

- If one of the recursive calls returned a model, then return this model. Otherwise (by recursion), we have for each $I \cup \{B_j(\overline{x}\Theta)\}$ an applicable rule of form $\forall \overline{y}_j\ \Phi_j(\overline{y}_j) \rightarrow \perp$.
- We will show that there is a way to obtain a lemma of form $\forall \overline{z}\ \Psi(\overline{z}) \rightarrow \perp$ that is applicable in I.

- If $Z(\overline{x})$ has form $\exists y\ B(\overline{x}, y)$, then for each $e \in E$, recursively call

$$\mathbf{search}^{+}(I \cup \{\ B(\overline{x}\Theta, e)\ \}, \mathcal{G})$$

and for one $\hat{e} \notin E$, recursively call

$$\mathbf{search}^{+}(I \cup \{\ B(\overline{x}\Theta, \hat{e})\ \}, \mathcal{G}).$$

- If one of the recursive calls returned a model, then return this model. Otherwise (by recursion), we have for each

$$I \cup \{\ B(\overline{x}\Theta, e)\ \}, \quad (e \in E)$$

and for

$$I \cup \{\ B(\overline{x}\Theta, \hat{e})\ \}, \quad \hat{e} \notin E,$$

an applicable lemma.

- We will show that there is a way to obtain a lemma of form $\forall \overline{z}\ \Psi(\overline{z}) \rightarrow \bot$, that is applicable in I.

Rules for Lemma Learning

A complete calculus can be obtained by the following three rules:

- Instantiation (followed by merging)
- Disjunction resolution.
- Existential resolution.

Lemma Factoring:

Let $\lambda =$

$$\forall \overline{x}\ A_1(\overline{x}) \wedge A_2(\overline{x}) \wedge \cdots \wedge A_p(\overline{x}) \wedge x_1 \not\approx x'_1 \wedge \cdots \wedge x_q \not\approx x'_q \rightarrow \bot,$$

be a lemma. Let Σ be a substitution of form $\{y := y'\}$. Then the following lemma is a factor of λ:

$$\forall \overline{x}\Sigma\ \ A_1(\overline{x}\Sigma) \wedge \cdots \wedge A_p(\overline{x}\Sigma) \wedge\ x_1\Sigma \not\approx x'_1\Sigma\ \wedge \cdots \wedge\ x_q\Sigma \not\approx x'_q\Sigma\ \rightarrow \bot.$$

Disjunction Resolution:

Let $\rho =$

$$\forall \overline{x} \; \Phi(\overline{x}) \to B_1(\overline{x}) \vee \cdots \vee B_q(\overline{x})$$

be a disjunctive formula.

Let $\lambda =$

$$\forall \overline{y} \; D_1(\overline{y}) \wedge \cdots \wedge D_r(\overline{y}) \wedge y_1 \not\approx y_1' \wedge \cdots \wedge y_s \not\approx y_s' \to \bot,$$

be a lemma, s.t. $B_1(\overline{x})$ and $D_1(\overline{y})$ are unifiable. Then the following formula is a disjunction resolvent of ρ and λ:

$$\forall \;\; \overline{x}\Sigma \;\; \overline{y}\Sigma \;\; \Phi(\overline{x})\Sigma \wedge$$

$$D_2(\overline{y})\Sigma \wedge \cdots \wedge D_r(\overline{y})\Sigma \;\wedge\; y_1\Sigma \not\approx y_1'\Sigma \;\wedge \cdots \wedge\; y_s\Sigma \not\approx y_s'\Sigma \;\to$$

$$B_2(\overline{x})\Sigma \vee \cdots \vee B_q(\overline{x})\Sigma.$$

Existential Resolution:

Let $\rho =$

$$\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \exists y\ B(\overline{x}, y)$$

be an existential formula.

Let $\lambda =$

$$\forall \overline{z}\ v \quad \Psi(\overline{z}) \wedge B(\overline{z}, v) \wedge v \not\approx z_1 \wedge \cdots \wedge v \not\approx z_s \rightarrow \bot,$$

be a lemma, s.t. $B(\overline{x}, y)$ and $B(\overline{z}, v)$ are unifiable and $v \notin \overline{z}$. Then the following formula is an existential resolvent of ρ and λ:

$$\forall\ \overline{x}\Sigma\ \ \overline{z}\Sigma \quad \Phi(\overline{x})\Sigma \wedge \Psi(\overline{z})\Sigma \rightarrow B(\overline{z}, z_1)\Sigma \vee \cdots \vee B(\overline{z}, z_s)\Sigma.$$

Providing some Evidence

Suppose we have $I = \; p(e_0), \;\; q(e_0)$.

Assume that the applicable rule is:

$$p(X) \rightarrow r(X) \vee s(X).$$

Assume that $p(e_0), \;\; q(e_0), \;\; r(e_0)$ has applicable rule

$$r(X) \rightarrow \bot.$$

Assume that $p(e_0), \;\; q(e_0), \;\; s(e_0)$ has applicable rule

$$q(X) \wedge s(X) \rightarrow \bot.$$

By disjunction resolution, one can obtain:

$$p(X) \wedge q(X) \rightarrow \bot.$$

Existential Resolution

The simplest form of existential resolution is:

From

$$p(X, Y) \rightarrow \exists z\ q(X, Y, z)$$

and

$$q(X, Y, Z) \wedge r(X, Y) \rightarrow \perp$$

derive

$$p(X, Y) \wedge r(X, Y) \rightarrow \perp.$$

Existential Resolution (2)

Now suppose we have

$$p(X, Y) \rightarrow \exists z \; q(X, Y, z)$$

and

$$q(X, Y, Z) \wedge Z \not\approx X \wedge r(X, Y) \rightarrow \bot.$$

The second rule refutes almost all possible choices for Z, except the case where $Z \approx X$.

Therefore, we must keep this possibility in the conclusion:

$$p(X, Y) \wedge r(X, Y) \rightarrow q(X, Y, X).$$

Existential Resolution (3)

Similarly,

$$p(X, Y) \to \exists z\; q(X, Y, z)$$

and

$$q(X, Y, Z) \wedge Z \not\approx X \wedge Z \not\approx Y \wedge r(X, Y) \to \bot$$

result in

$$p(X, Y) \wedge r(X, Y) \to q(X, Y, X) \vee q(X, Y, Y).$$

Providing Evidence for Existential Resolution

Suppose that we have $I = p(e_0)$.

Assume that the applicable rule is $\rightarrow \exists y \; q(y)$.

Assume that $p(e_0), \;\; q(e_0)$ has applicable rule

$$p(X) \wedge q(X) \rightarrow \bot.$$

Assume that $p(e_0), \;\; q(e_1)$ has applicable rule

$$p(X) \wedge q(Y) \wedge X \not\approx Y \rightarrow \bot.$$

Existential resolution gives

$$p(X) \rightarrow q(X).$$

Disjunction resolution results in

$$p(X) \rightarrow \bot.$$

Theorem: The calculus does what it is supposed to do.

That is: On every choice point, it derives a new closing lemma that closes the interpretation before the choice point, using the closing lemmas obtained from the recursive calls.

Implementation

We have an implementation of this calculus, which is called geo. it took part in this year's CASC. It solved:

FOF: 73 out of 150,
CNF: 45 out of 150,
SAT: 51 out of 100,
UEQ: 2 out of 100.

This is not bad for a first time, but there is still a lot of work to do.

Matching

The most interesting part of geo is its matching algorithm:

Matching: Given an interpretation I and a set of geometric formulas $\mathcal{G}$, find a geometric formula $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow Z(\overline{x})$, that is applicable on I with ground substitution Θ.

In practice, this is only important for $Z(\overline{x}) = \bot$.

First Solution: Naive implementation, using backtracking.

Performance $\Rightarrow$ hopeless.

Second solution: Observe that in nearly all cases, rules

$$\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \bot$$

are 'near splittable'. This means that $\Phi(\overline{x})$ can be partitioned into two parts $\Phi_1(\overline{x}_1, \overline{y}) \cup \Phi_2(\overline{x}_2, \overline{y})$, such that $\overline{y}$ is small in comparison to $\overline{x}$.

Store substitutions on $\overline{y}$ and remember whether they result in a matching.

Performance $\Rightarrow$ much better, but excessive memory use.

Matching Algorithm

Definition: A substitution lemma is an object of form $\lambda \to \perp$, where λ is a ground substitution.

Definition: A substition clause is an object of form $\Theta_1 \vee \cdots \vee \Theta_p$, where $\Theta_1, \ldots, \Theta_p$ are ground substitutions with the same domain.

A state of the matching algorithm consists of a triple (Θ, C, Λ), where

- Θ is a ground substitution.
- C is a set of substitution clauses.
- Λ is a set of substitution lemmas.

The algorithm $M(\Theta, C, \Lambda)$ returns either a ground substitution Θ that satisfies all the clauses in C or a substitution lemma $\lambda \rightarrow \bot$ for which $\lambda \subseteq \Theta$.

- If there exists a substitution lemma $\lambda \rightarrow \bot$, s.t. $\lambda \subseteq \Theta$, then $M(\Theta, C, \Lambda)$ returns this lemma.
- If for each $\Theta_1 \vee \cdots \vee \Theta_p \in C$, there is a j s.t. $\Theta_j \subseteq \Theta$, then return $M(\Theta, C, \Lambda)$ returns Θ.

- Let $c \in C$ be a substitution clause of form $\Theta_1 \vee \cdots \vee \Theta_p$. Let $\Sigma_1 \vee \cdots \vee \Sigma_n$ be the part of c that is consistent with Θ.
  Let $\Xi_1 \vee \cdots \vee \Xi_m$ be the part of c that is inconsistent with Θ.
  (So we have $n + m = p$)

  For each Σ_i, compute $\sigma_i := M(\Theta + \Sigma_i,\ C,\ \Lambda)$.
  If one of the σ_i is a substitution, then return σ_i.

  Otherwise, all σ_i are lemmas of form $\lambda_i \rightarrow \bot$. If one of the λ_i has $\lambda_i \subseteq \Theta$, then return $\lambda_i \rightarrow \bot$.

  Otherwise, let Θ' be a minimal subset of Θ that is inconsistent with all of $\Xi_1, \ldots, \Xi_m$. For $1 \leq i \leq n$, define $\lambda_i' = \lambda_i \cap \Theta$.
  Then $M(\Theta, C, \Lambda) \ = \ (\lambda_1' \cup \cdots \cup \lambda_n') \cup \Theta' \rightarrow \bot$.

This matching algorithm gives acceptable performance. Hardest instances produce about 50000 lemmas.

Most matches are computed < 0.01 seconds.

Optimizations of the Calculus

At this moment, I have two optimizations of the calculus:

Subsumption (As usual) If there are two lemmas $\forall \overline{x}\ \Phi(\overline{x}) \rightarrow \bot$, and $\forall \overline{y}\ \Psi(\overline{y}) \rightarrow \bot$, and there is a substitution Σ, s.t. $\Phi(\overline{x})\Sigma \subseteq \Psi(\overline{y})$, then $\forall \overline{y}\ \Psi(\overline{y}) \rightarrow \bot$ can be deleted.

Functional Reduction If there is a lemma of form $\forall \overline{x}\ Y_1\ Y_2\ \Phi(\overline{x}) \wedge F(\overline{x}, Y_1) \wedge F(\overline{x}, Y_2) \rightarrow \bot$, and the only positive occurrence of F is in a rule of form $\forall \overline{z}\ \Psi(\overline{z}) \rightarrow \exists y\ F(\overline{z}, y)$, then Y_1 and Y_2 can be unified.

Conclusions, Future Work

- We gave a new calculus, which is somewhat similar to resolution, and which is refutationally complete for first-order logic.
- Since the algorithm provides an implicit completeness proof, this calculus could be used for saturation-based theorem proving.
- But we do not recommand this: The calculus is intended to be used in combination with the model search algorithm.
- In the implementation, understand which lemmas should be forgotten. Find good heuristics. Develope an intuition of how it searches, and what the proofs mean.
- Extend calculus? (theories, well-behaved infinite models)