CS389L: Automated Logical Reasoning

Lecture 8: Introduction to Theorem Proving

Işıl Dillig

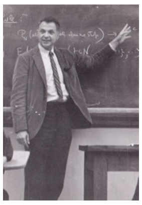
Review

- What are some decidable fragments of FOL?
- What is compactness?
- What is a property that cannot be expressed in FOL?

First-Order Theorem Provers

- A first-order theorem prover is a computer program that proves the validity of formulas in first-order logic.
- Since validity in FOL is only semi-decidable, first-order theorem provers are not guaranteed to terminate
- Despite this limitation, many automated theorem provers exist and are useful: Vampire, SPASS, Otter, ...
- There are even annual competitions between these theorem provers! (just Google "CADE ATP competition")

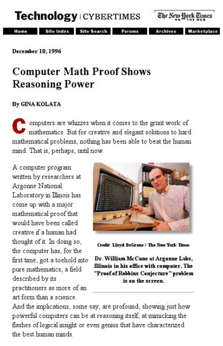
Theorem Provers and Mathematical Theorems



- First-order theorem provers have been used to prove some mathematical theorems not previously proven by humans.
- Robbins conjecture (1933): Mathematician Herbert Robbins conjectured that a group of axioms he came up with are equivalent to boolean algebra.
- Neither he nor anyone else could prove this for decades.

Robbins Conjecture and Automated Theorem Proving

- 1996: Conjecture eventually proven by first-order theorem prover EQP after 8 days of search!
- That a computer can prove theorems that humans could not was shocking
- The automated proof of Robbins conjecture even appeared as New York Times article!
- Not the only success story: Otter used by mathematician Ken Kunnen to prove results in quasi-group theory



Technology | CYBERTIMES The New York Times ON THE WEB

Home | Site Index | Site Search | Forums | Archives | Marketplace

December 10, 1996

Computer Math Proof Shows Reasoning Power

By GINA KOLATA

Computers are whizzes when it comes to the grunt work of mathematics. But for creative and elegant solutions to hard mathematical problems, nothing has been able to beat the human mind. That is, perhaps, until now.

A computer program written by researchers at Argonne National Laboratory in Illinois has come up with a major mathematical proof that would have been called creative if a human had thought of it. In doing so, the computer has, for the first time, got a toehold into pure mathematics, a field described by its practitioners as more of an art form than a science.

Credit: Lloyd DeGrane / The New York Times

Dr. William McCune at Argonne Labs, Illinois in his office with computer. The "Proof of Robbins Conjecture" problem is on the screen.

And the implications, some say, are profound, showing just how powerful computers can be at reasoning itself, at mimicking the flashes of logical insight or even genius that have characterized the best human minds.

Overview

- Today's lecture and next lecture: Discuss basic principles underlying first-order theorem provers
- The basis underlying all theorem provers today is the principle of first-order resolution
- First-order theorem provers prove formulas unsatisfiable by showing there is a resolution refutation for that formula

Recall: Propositional Resolution

- Recall: Resolution in propositional logic:

$$C_1: \ (l_1 \vee \ldots p \ldots \vee l_k) \qquad C_2: \ (l'_1 \vee \ldots \neg p \ldots \vee l'_n)$$

- Propositional resolution: Deduction of a new clause C_3, called resolvent:

$$C_3: \ (l_1 \vee \ldots \vee l_k \vee l'_1 \vee \ldots \ldots \vee l'_n)$$

- First-order resolution is the same basic principle, but a little bit more involved
  - How to obtain clauses given FOL formula?
  - How do we deal with predicates containing syntactically different terms?

First-Order Resolution Prerequisites

- To perform resolution in first-order logic, we need two new ingredients:
  1. Unification: Which expressions can be made identical?
  2. Clausal form: A new normal form for FOL
- Start with unification; then talk about clausal form

Unification

- Unification: problem of determining if two expressions can be made identical by appropriate substitutions for their variables
- Substitution: finite mapping from variables to terms
- Example: Can expressions $p(x)$ and $p(a)$ be unified?
- Can $p(a)$ and $p(b)$ be unified?
- We'll write $e\sigma$ to denote the application of substitution σ to expression e
- What is $p(x)[x \mapsto a]$?

Unification

- A substitution is a unifier for two expressions e and e' if $e\sigma$ is syntactically identical to $e'\sigma$
- Two expressions e and e' are unifiable iff they have a unifier; otherwise non-unifiable.
- Example: Are $p(x, y)$ and $p(a, v)$ unifiable?
- A unifier:
- Example 2: Are $p(x, x)$ and $p(a, b)$ unifiable?
- Example 3: Are $p(x)$ and $p(f(x))$ unifiable?

Non-Uniqueness of Unifiers

- If two expressions are unifiable, they don't necessarily have a unique unifier.
- Example: $p(x, y)$ and $p(a, v)$
- Unifier 1: $[x \mapsto a, y \mapsto b, v \mapsto b]$
- Unifier 2: $[x \mapsto a, y \mapsto v]$
- Unifier 3: $[x \mapsto a, y \mapsto f(b), v \mapsto f(b)]$
- But some unifiers are more desirable than others ...

Composing Substitutions

- To explain what it means for one unifier to be better than another, we define the composition of substitutions.
- Composition of two substitutions σ and δ is written $\sigma\delta = \sigma'$
- The composition $\sigma\delta$ of substitutions σ and δ is obtained by:
  1. applying δ to the range of σ
  2. add to σ all mappings $x \mapsto t$ from δ where $x \notin dom(\sigma)$.

Composing Substitutions Examples

- What is $[x \mapsto a, y \mapsto z][z \mapsto b]$?
- What is $[x \mapsto a, y \mapsto f(z, g(w))][z \mapsto 1, w \mapsto 2]$?
- Let

$$\sigma = [x \mapsto a, y \mapsto f(u), z \mapsto v]$$
$$\delta = [u \mapsto d, v \mapsto e, z \mapsto g]$$

  What is $\sigma\delta$?

Generality of Unifiers

- We prefer unifiers that are as general as possible.
- A unifier σ is at least as general as unifier σ' if there exists another substitution δ such that $\sigma\delta = \sigma'$
- Intuition: σ more general than σ' if σ' can be obtained from σ through another substitution
- We say σ more general than σ' if σ is at least as general as σ' but not the other way around
- Which unifier is more general? $\sigma = [x \mapsto a, y \mapsto v]$ or $\sigma' = [x \mapsto a, y \mapsto f(c), v \mapsto f(c)]$?
- Which unifier is more general? $\sigma = [x \mapsto a, y \mapsto z]$ or $\sigma' = [x \mapsto a, y \mapsto w]$?

Most General Unifiers

- σ is a most general unifier (mgu) of expressions e, e' iff σ is at least as general as any other unifier of e and e'.
- Intuition: A unifer is most general if it only contains mappings necessary to unify, but nothing extra!
- Consider again $p(x, y)$ and $p(a, v)$.
- Is $[x \mapsto a, y \mapsto b, v \mapsto b]$ an mgu?
- Is $[x \mapsto a, y \mapsto v]$ an mgu?
- Is $[x \mapsto a, y \mapsto v, v \mapsto y]$ an mgu?
- If two expressions e and e' are unifiable, then their mgu is unique modulo variable renaming

Algorithm to Compute MGU

- We'll now give an algorithm to find most general unifiers (Robinson's algorithm, 1965)
- Function `find_mgu(e, e')` takes expressions e, e' and returns substitution σ that is mgu of e, e' or $\perp$
- Case 1: $e = e'$. Then $\sigma = [\,]$
- Case 2: e is variable x. If e' does not contain x then $[x \mapsto e']$, otherwise $\perp$
  - Containment check referred to as occurs check; disallows infinite terms as a solution
- Case 3: e' is variable $y \Rightarrow$ symmetric to case 2
- Case 4: e or e' is a constant. Return $\perp$

Algorithm to Compute MGU, continued

- Case 5: $e = \tau(e_1, \ldots, e_k)$.
  1. If $e' \neq \tau(e'_1, \ldots, e'_k)$, then $\perp$
  2. Otherwise result of unifying $[e_1 \ldots e_k]$ and $[e'_1 \ldots e'_k]$
- Case 6: e is expression list $[h\ T]$.
  1. If e' is not expression list of the form $[h'\ T']$, return $\perp$.
  2. Let $\sigma = \texttt{find\_mgu}(h, h')$.
  3. Apply σ to T, T'
  4. Recursively compute MGU σ' for σT and $\sigma T'$
  5. Return composition of σ and σ'

Example of Computing MGUs

- Apply algorithm to find mgu for $p(f(x), f(x))$ and $p(y, f(a))$
- Predicates match; unify the arguments.
- Unify first arguments $f(x)$ and y
- Result:
- Apply unifier to second arguments $f(x)$ and $f(a)$ (unchanged)
- Then, unify $f(x)$ and $f(a)$:
- Compose $[y \mapsto f(x)]$ and $[x \mapsto a]$
- Final result:

Another Example

- Apply algorithm to find mgu for $p(x, x)$ and $p(y, f(y))$
- Predicates match; unify the arguments.
- Unify first arguments x and y; result:
- Apply unifier to second arguments x and $f(y)$:
- Now unify y and $f(y)$:
- Thus $p(x, x)$ and $p(y, f(y))$ not unifiable

Complexity of unification

- Robinson's algorithm has worst-case complexity, but only triggered in "pathological" cases
- There are almost-linear time unification algorithms, but Robinson's algorithm still widely used

First-Order Resolution Ingredients

- Recall: Resolution in FOL requires two new ingredients: unification and clausal form
- Next, we'll define clausal form
- A formula in FOL in said to be in clausal form it obeys following syntactic restrictions:
  1. Formula should be of the form $\forall x_1, \ldots, x_k.\ F(x_1, \ldots x_k)$ (i.e., only universally quantified variables)
  2. The inner formula $F(x_1, \ldots, x_k)$ should be in CNF

The Bad and The Good News

- Bad News:
  In general, if ϕ is the original formula, there may not be an equivalent formula ϕ' that is of this form
- Good News:
  But we can always find an equi-satisfiable formula ϕ'' that is of this form
- Since we are trying to determine satisfiability of ϕ, this is good enough ...

Converting Formulas to Equisatisfiable Clausal Form

Given formula ϕ, there are five steps to convert it to equisatisfiable clausal form:

1. Make sure there are no free variables in ϕ
2. Convert resulting formula to Prenex normal form
3. Apply skolemization to remove existentially quantified variables (resulting formula called Skolem Normal Form)
4. Since formula obtained after step 3 is of the form $\forall x_1, \ldots, x_k.\ F(x_1, \ldots x_k)$, convert inner formula F to CNF
5. Since all variables are universally quantified, drop explicit quantifiers and write formula as set of clauses

Step 1: Removing Free Variables

- Suppose a formula ϕ contains free variable x
- How can we construct a formula ϕ' such that x is no longer free and ϕ' is equisatisfiable to ϕ?
- ϕ is satisfiable iff there exists some $o \in U$ under which $U, I, \{x \mapsto o\} \models \phi$
- But this is the same as saying ϕ is satisfiable iff $U, I \models \exists x.\phi$ for some U, I
- Thus, to perform step 1 of transformation, existentially quantify all free variables of ϕ

Prenex Normal Form

- A formula is in prenex normal form (PNF) if all of its quantifiers appear at the beginning of formula:

$$Qx_1, \ldots Qx_n.\ F(x_1, \ldots, x_n)$$

where F is quantifier-free and $Q \in \{\forall, \exists\}$

- Is $\forall x.\exists y.(p(x,y) \rightarrow q(x))$ in PNF?
- What about $\forall x.((\exists y.p(x,y)) \rightarrow q(x))$ in PNF?

Step 2: Conversion to Prenex Normal Form

- Step2a: Convert to NNF.
- Conversion to NNF is just like in propositional logic, but need new equivalences for distributing negation over quantifiers:

$$\neg\forall x.\phi \quad \Leftrightarrow \quad \exists x.\neg\phi$$
$$\neg\exists x.\phi \quad \Leftrightarrow \quad \forall x.\neg\phi$$

- Step 2b: Rename quantified variables as necessary so no two quantified variables have the same name.
- Step 2c: Move quantifiers to front of formula $Q_1x_1, \ldots, Q_nx_n.F'$ such that if Q_j is in the scope of quantifier Q_i, then $i < j$.
- Claim: Formula in PNF is equivalent to original formula.

Conversion to PNF Example

- Convert formula to PNF:

$$\forall x.\neg(\exists y.p(x,y) \land p(x,z)) \lor \exists y.p(x,y)$$

Step 3: Skolemization

- After converting formula to PNF, we want to remove all existential quantifiers
- Skolemization produces equisatisfiable formula without existential quantifiers
- Suppose an existentially quantified variable y appears in the scope of quantifiers $x_1, \ldots, x_k$
- Skolemization: replaces y with function term: $f(x_1, \ldots, x_n)$ where f is a fresh function symbol
- This new function f called Skolem function
- What happens if y is not in scope of any quantifiers?

Skolemization: Intuition I

- Consider formula $\exists x.F$
- We know there is some object for which F holds, but we don't want to make any assumptions about this object
- Thus, we replace x with a fresh object constant c in F
- The formula $F[c/x]$ is equisatisfiable to $\exists x.F$, but not equivalent

Skolemization: Intuition II

- However, if existential variable x is in scope of universally quantified variables, we can't replace it with object constant
- Consider formula: $\forall x.\exists y.hates(x,y)$
- What does this formula say?
- Now, let's replace y with object constant c: $\forall x.hates(x,c)$
- What does this formula say?
- Clearly, very different meaning!
- Want to capture that two people can hate different people $\Rightarrow$ introduce function constant

Skolem Normal Form

- The formula after performing skolemization looks like this:

$$\forall x_1, \ldots, \forall x_n.\ F(x_1, \ldots, x_n)$$

- This form is called Skolem Normal Form
- Resulting formula not equivalent to original formula, but equisatisfiable

Conversion to Clausal Form Example

- Convert formula to clausal form:

$$\forall y.(p(y) \land \neg(\forall z.(r(z) \rightarrow q(y, z, w))))$$

- Step 1: Remove free variables:

$$\exists w.\forall y.(p(y) \land \neg(\forall z.(r(z) \rightarrow q(y, z, w))))$$

- Step 2a: Convert to NNF (necessary for PNF):

$$\exists w.\forall y.(p(y) \land \neg(\forall z.(\neg r(z) \lor q(y, z, w)))) \quad \text{remove} \rightarrow$$
$$\exists w.\forall y.(p(y) \land (\exists z.(r(z) \land \neg q(y, z, w)))) \quad \text{push negations}$$

- Step 2b: Move quantifiers out (necessary for PNF):

$$\exists w.\forall y.\exists z.(p(y) \land ((r(z) \land \neg q(y, z, w))))$$

Conversion to Clausal Form Example, continued

- In Prenex Normal Form:

$$\exists w.\forall y.\exists z.(p(y) \land ((r(z) \land \neg q(y, z, w))))$$

- Step 3a: Now, skolemize w (easiest to start outside):

$$\forall y.\exists z.(p(y) \land ((r(z) \land \neg q(y, z, c))))$$

- Step 3b: Skolemize z:

$$\forall y.(p(y) \land ((r(f(y)) \land \neg q(y, f(y), c))))$$

Conversion to Clausal Form Example, continued

- In Skolem Normal Form:

$$\forall y.(p(y) \land ((r(f(y)) \land \neg q(y, f(y), c))))$$

- Step 4: Convert inner formula to CNF (already in CNF)
- Step 5: Drop universal quantifiers:

$$(p(y) \land ((r(f(y)) \land \neg q(y, f(y), c))))$$

- Step 6: Finally, write formula as a set of clauses

$$\{p(y)\}, \{(r(f(y))\}, \{q(y, f(y), c)\}$$

- This formula is now in clausal form

Summary

- Today: Talked about two necessary ingredients for first-order resolution:
  1. Unification
  2. Clausal form
- Next lecture: First-order resolution and theorem provers