Computational Logic

Automated Theorem Proving

Damiano Zanardini

UPM European Master in Computational Logic (EMCL)
School of Computer Science
Technical University of Madrid
damiano@fi.upm.es

Academic Year 2008/2009

Introduction

A recipe

The ingredients

- first-order logic with *equality*
- yet another inference rule: *paramodulation*

The problem

- the Robbins problem: that *every Robbins algebra is a Boolean algebra*

The tool

- the EQP theorem prover

Equality

Example

- axioms:
  - $even(sum(twoSquared, b))$
  - $twoSquared = four$
  - $\forall x(zero(x) \rightarrow difference(four, x) = sum(four, x))$
  - $zero(b)$
- conjecture:
  - $even(difference(twoSquared, b))$
- the conjecture could seem like a logical consequence of the axioms
- however, this is due to the fact that a human *knows* what equality means

Equality

A non-standard interpretation

$$
\begin{array}{rcl@{\qquad}rcl}
D & = & \{cat, dog\} & difference(cat, cat) & = & dog \\
b & = & cat & difference(cat, dog) & = & cat \\
twoSquared & = & cat & difference(dog, cat) & = & cat \\
four & = & cat & difference(dog, dog) & = & cat \\
sum(cat, cat) & = & cat & (cat{=}cat) & = & \mathbf{t} \\
sum(cat, dog) & = & cat & (cat{=}dog) & = & \mathbf{f} \\
sum(dog, cat) & = & cat & (dog{=}cat) & = & \mathbf{t} \quad (!) \\
sum(dog, dog) & = & cat & (dog{=}dog) & = & \mathbf{f} \quad (!) \\
even(cat) & = & \mathbf{t} & zero(cat) & = & \mathbf{t} \\
even(dog) & = & \mathbf{f} & zero(dog) & = & \mathbf{f}
\end{array}
$$

This interpretation satisfies the axioms but *not* the conjecture

Equality

Equality axioms

In order to establish the above logical consequence, it is necessary to add the behavior of $= /2$ as a set of non-logical axioms

- reflexivity: $\forall x(x = x)$
- simmetry: $\forall x \forall y(x = y \rightarrow y = x)$
- transitivity: $\forall x \forall y \forall z((x = y \wedge y = z) \rightarrow x = z)$
- function substitution: if $x = y$, then $f(x) = f(y)$
  - for every argument of every function: Ex. $\forall x \forall y \forall z(x = y \rightarrow sum(x, z) = sum(y, z))$
- predicate substitution: if $x = y$ and $p(x)$ is true, then $p(y)$ is also true
  - for every argument of every predicate: Ex. $\forall x \forall y(x = y \rightarrow (even(x) \rightarrow even(y)))$

Paramodulation (Robinson-Wos, 1969)

Paramodulants

- paramodulation is an inference rule which generates all *equal* versions of clauses modulo the equality information
- it does the job of all equality axioms except reflexivity
- the *paramodulant* is the resulting clause

Paramodulation (Robinson-Wos, 1969)

Formal definition

- two parent clauses: *from* clause F and *input* clause I
- F must contain a positive equality literal E

$$F \quad \equiv \quad (t_1{=}t_2) \vee C$$

- one of the arguments of E must unify (with $MGU\ \alpha$) with a subterm t of I

$$I \quad \equiv \quad D[t] \quad \text{and} \quad (\alpha = MGU(t_1, t) \quad \text{or} \quad \alpha = MGU(t_2, t))$$

- t is replaced in I by the *other* argument of E

$$I \quad \rightsquigarrow \quad I(t/t_2) \quad \text{or} \quad I \quad \rightsquigarrow \quad I(t/t_1)$$

- α is applied to the new I and the remaining part of F

$$P \quad \equiv \quad (C \vee I(t/t_2))\alpha \quad \text{or} \quad P \quad \equiv \quad (C \vee I(t/t_1))\alpha$$

Paramodulation (Robinson-Wos, 1969)

Example

- $F \quad \equiv \quad C \vee (t_1{=}t_2) \quad \equiv \quad p(x,y) \quad \vee \quad (f(x){=}g(a))$
- $I \quad \equiv \quad p(g(z), f(h(f(a), f(b)))) \quad \vee \quad q(f(a))$
- $t_1 \equiv f(x)$ unifies with $t \equiv f(h(f(a), f(b)))$ with MGU

$$\alpha = \{x/h(f(a), f(b))\}$$

- $I' \quad \equiv \quad I(t/t_2) \quad \equiv \quad p(g(z), g(a)) \quad \vee \quad q(f(a))$
-
$$\begin{aligned} P \quad &\equiv \quad (C \vee I')\alpha \\ &\equiv \quad (p(x,y) \quad \vee \quad p(g(z), g(a)) \quad \vee \quad q(f(a))) \quad (\{x/h(f(a), f(b))\}) \\ &\equiv \quad p(h(f(a), f(b)), y) \quad \vee \quad p(g(z), g(a)) \quad \vee \quad q(f(a)) \end{aligned}$$

Paramodulation (Robinson-Wos, 1969)

Lemma (Correctness)

P is a logical consequence of $F \wedge I$

Proof.

1. suppose $\neg P$, i.e., $\neg((C \vee I')\alpha)$
2. $\neg(I'\alpha)$ (from ❶ and $\vee$ elimination)
3. $\neg(I\alpha)$ (from ❷ and $I\alpha = I'\alpha$ (definition of α))
4. $\neg I$ (from ❸ and properties of substitutions)
5. $\neg(F \wedge I)$ (from ❹)

Paramodulation (Robinson-Wos, 1969)

Real-life example

- $I \;\equiv\; n(n(n(x)+y)+n(x+y)) = y$
- $F \;\equiv\; n(n(n(x)+y)+n(x+y)) = y$
- (renaming) $I \;\equiv\; n(n(n(x')+y')+n(x'+y')) = y' \quad\rightsquigarrow\quad t$
- (renaming) $F \;\equiv\; n(n(n(x'')+y'')+n(x''+y'')) = y'' \quad\rightsquigarrow\quad t_1$
- $\alpha \;=\; \{\; x'/(n(x'')+y''),\;\; y'/(n(x''+y''))\; \}$
- $I' \;\equiv\; n(y''+n(x'+y')) = y'$
- $$\begin{aligned} P &\equiv I'\alpha \\ &\equiv n(y''+n(n(x'')+y''+n(x''+y''))) = n(x''+y'') \\ &\equiv n(n(n(x''+y'')+n(x'')+y'')+y'') = n(x''+y'') \end{aligned}$$

EQP and the Robbins problem

A bit of history

Mathematicians have long struggled against a difficult algebra problem: that the definition of a *Boolean algebra* is equivalent to that of a *Robbins algebra* (from Herbert Ellis Robbins (1915-2001))

- one direction (that *every Boolean algebra is a Robbins algebra*) is easy
- but the other one (that *every Robbins algebra is a Boolean algebra*) is extremely difficult

EQP and the Robbins problem

A partial result

- in 1979, Larry Wos told his colleague Steve Winker to attack the problem by *strengthening the hypotheses*
- i.e., find *conditions* which, if true, would solve the problem
  - Winker: *what does such an attack give me as a mathematician?*
  - Wos: *nothing; but as a gambler it tells you a lot*
- in 1990, Steve Winker showed that *each* of two conditions (the *Winker conditions*) are sufficient in order to make a Robbins algebra Boolean
- the proof was by hand, *with insight from theorem prover searches*
- lately, *automated* proofs were found (1992 for the first condition, 1996 for the second)
- yet, the problems remains: does any Robbins algebra satisfy at least one of the Winker conditions?

EQP and the Robbins problem

Boolean axioms

commutativity	$x + y = y + x$	$x \cdot y = y \cdot x$
associativity	$(x + y) + z = x + (y + z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
zero	$0 + x = x + 0 = x$	$0 \cdot a = a \cdot 0 = 0$
one	$1 + a = a + 1 = 1$	$1 \cdot a = a \cdot 1 = a$
distributivity	$a + b \cdot c = (a + b) \cdot (a + c)$	$a \cdot (b + c) = a \cdot b + a \cdot c$
absorption	$x \cdot (x + y) = x + x \cdot y = x$	
complementation	$\forall x \exists y (x \cdot y = 0 \wedge x + y = 1)$ $x \cdot n(x) = 0,\ x + n(x) = 1$	

Robbins axioms

commutativity	$x + y = y + x$
associativity	$(x + y) + z = x + (y + z)$
Robbins' axiom	$n(n(n(x) + y) + n(x + y)) = y$

EQP and the Robbins problem

How the problem is formulated

Given the Robbins axiom (and the equality axioms EQ), is it possible to prove the second Winker condition?

- this would demostrate that every Robbins algebra is a Boolean algebra
- premises

$$\begin{array}{ll} (1) & x + y = y + x \\ (2) & (x + y) + z = x + (y + z) \\ (3) & n(n(n(x) + y) + n(x + y)) = y \end{array}$$

- conclusion (second Winker condition)

$$\exists x \exists y (n(x + y) = n(x))$$

- negated conclusion

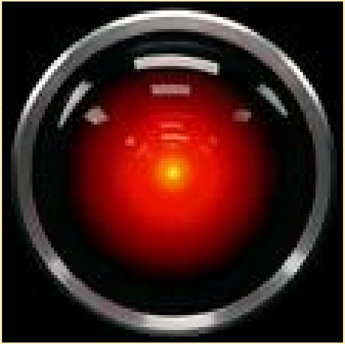
$$(4) \quad n(x + y) \neq n(x)$$

- is the set $\{(1), (2), (3)\} \cup EQ \cup \{(4)\}$ satisfiable?

EQP and the Robbins problem

When *machines do it better*

- not only HAL...



- *became "operational" on January 12, 1997*

EQP and the Robbins problem

When *machines do it better*

- ...or Deep(er) Blue

- on May 11th 1997, won a six-game match by two wins to one with three draws against world champion Garry Kasparov

EQP and the Robbins problem

When *machines do it better*

- in September 1996, William McCune startled Wos by bringing up the Robbins problem, asserting *I think we can get it*
- McCune suspected that a new program he had developed called EQP (for *equational prover*) just might do the trick...
- ...but confesses he was as amazed as anyone when, eight days later, the computer spewed out a proof
- hand-checking by McCune and several outside mathematicians confirmed that it was *indisputably correct*
- the proof took 678232.2 seconds, and generated 18K formulæ
- however, the final proof only consisted of 17 formulæ

EQP and the Robbins problem

The proof

```
----- EQP 0.9, June 1996 -----
The job began on eyas09.mcs.anl.gov, Wed Oct 2 12:25:37 1996
UNIT CONFLICT from 17666 and 2 at 678232.20 seconds.

---------------- PROOF ----------------

2 (wt=7) [] -(n(x+y) = n(x)).
3 (wt=13) [] n(n(n(x)+y) + n(x+y)) = y.
5 (wt=18) [para(3,3)] n(n(n(x+y)+n(x)+y)+y) = n(x+y).
6 (wt=19) [para(3,3)] n(n(n(n(x)+y)+x+y)+y) = n(n(x)+y).
...
17666 (wt=33) [para(24,16426),demod([17547])]
    n(n(n(x)+x)+n(n(x)+x)+x+x+x+x) = n(n(n(x)+x)+x+x+x).
------------ end of proof -------------
```

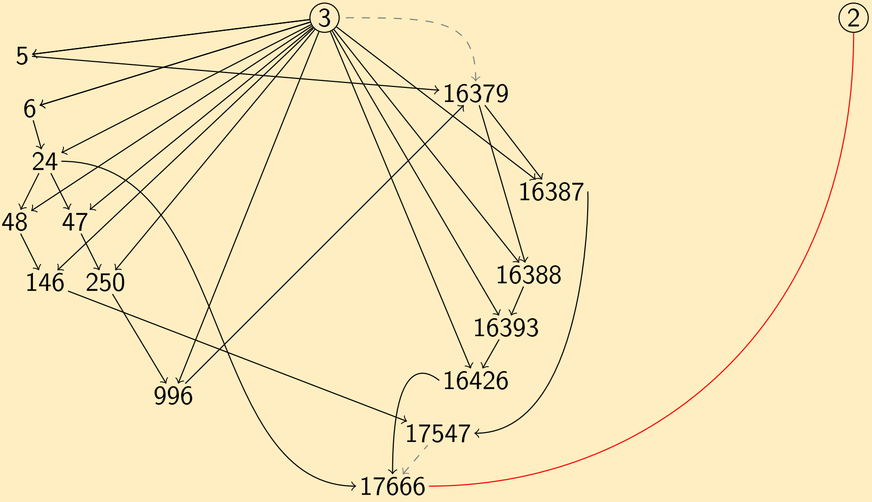
The proof

```
----- EQP 0.9, June 1996 -----
The job began on eyas09.mcs.anl.gov, Wed Oct 2 12:25:37 1996
UNIT CONFLICT from 17666 and 2 at 678232.20 seconds.

---------------- PROOF ----------------

2 (wt=7) [] -(n(x+y) = n(x)).
3 (wt=13) [] n(n(n(x)+y) + n(x+y)) = y.
5 (wt=18) [para(3,3)] n(n(n(x+y)+n(x)+y)+y) = n(x+y).
6 (wt=19) [para(3,3)] n(n(n(n(x)+y)+x+y)+y) = n(n(x)+y).
...
17666 (wt=33) [para(24,16426),demod([17547])]
    n(n(n(x)+x)+n(n(x)+x)+x+x+x+x) = n(n(n(x)+x)+x+x+x).
------------ end of proof -------------
```

- conflict: $x = n(n(x) + x) + x + x + x$ $\quad y = n(n(x) + x) + x$

EQP and the Robbins problem

The derivation



EQP and the Robbins problem

According to senior Argonne mathematician Larry Wos

- computers beating chess masters like Garry Kasparov may draw bigger headlines, but solving the Robbins conjecture is a far bigger deal
- if we're interested in track and we can't win a race against the high school kids, how the hell are we going to get on the Olympic team? And now we've finally reached that level
- people don't want to think any machine can do something they can't do. They don't want to feel like they're becoming obsolete. They want to do it themselves
- we don't just prove theorems. We look at conjectures, we design circuits, we solve puzzles, we prove properties of other programs
- anyway, why would you want to program a computer to be vicious, crabby, selfish, and inconsiderate, when humans do all of those things so very well?

Other ATP resources

Provers

- ACL2, Agda, Carine, Coq, DCTP, E, Gandalf, Isabelle, Jape, KeY, Larch, LCF, Lean, Matita, Otter, PhoX, Prover9, SETHEO, Tau, Twelf, Uclid, Vampire, Waldmeister...

Tests

- the Thousands of Problems for Theorem Provers (TPTP) Problem Library: `http://www.tptp.org/`

Other ATP resources

Contests

CADE ATP System Competition (CASC)

- FOF (*First-order form non-propositional theorems (axioms with a provable conjecture)*): Vampire won 8 times
- CNF (*Mixed clause normal form really non-propositional theorems (unsatisfiable clause sets)*) : Vampire won 9 times
- SAT (*Clause normal form really non-propositional non-theorems (satisfiable clause sets)*): Gandalf won 5 times
- EPR (*Effectively propositional clause normal form theorems and non-theorems (clause sets)*): DCTP won 3 times
- UEQ (*Unit equality clause normal form really non-propositional theorems (unsatisfiable clause sets)*): Waldmeister won 12 times

Related problems

Proof verification

- or *proof checking*
- easier, decidable if every step can be checked by a primitive recursive function

Interactive provers

- a human user provides hints to the system
- somehow between proving and checking

Related problems

Model checking

- a process is considered theorem proving if it consists of a traditional proof obtained by axioms and inference rules
- from *Model Checking vs. Theorem Proving: A Manifesto* (Halpern-Vardi)

  > *We argue that rather than representing an agent's knowledge as a collection of formulas, and then doing theorem proving to see if a given formula follows from an agent's knowledge base, it may be more useful to represent this knowledge by a semantic model, and then do model checking to see if the given formula is true in that model. We discuss how to construct a model that represents an agent's knowledge in a number of different contexts, and then consider how to approach the model-checking problem.*

- brute-force enumeration of many possible states
- yet, actual implementation are far from being brute-force

Related problems

Hybrid theorem proving

- model checking as an inference rule

Programs

- programs which prove a particular theorem, with a (usually informal) proof that termination with a certain result implies the theorem
- works on huge (non-surveyable) proofs
  - four color theorem (1976, later ATP proof in 2005, still huge)
  - the game *four in a line*: first player wins

Other uses

Industrial uses

- mostly concentrated in integrated circuit design and verification
- since the Pentium FDIV bug (1994), the complicated floating point units of modern microprocessors have been designed with extra scrutiny
- in the latest processors from AMD, Intel, and others, ATP has been used to verify that division and other operations are correct