TOPICS 2009

AUTOMATED REASONING

Krysia Broda
Presented by Rob Craven

Formal Deduction using a Computer

Why use a computer to prove things?

- It's quite hard for us to prove things
- Some problems are just too boring

==> We make mistakes,
whereas a correct program would not.

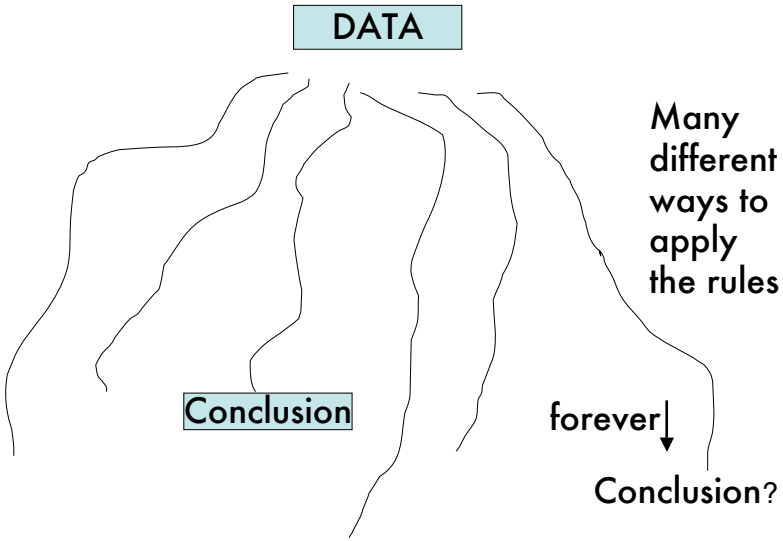
How can we do it?

1. Write down the problem in a formal notation
   (e.g. write the data and conclusion in logic)

2. Encode the allowed reasoning rules

3. While conclusion-is-not-derived-from-data
apply a rule to current data to derive more data

Why might this not work very well?



Instead:

Use a restricted notation called clausal form

Enables a mix of
reasoning backwards from the goal, and
reasoning forwards from the data

Requires only one inference rule – Resolution

Clausal Form
(also called Conjunctive Normal Form)

culprit(dolly) ∨ culprit(frances) ∨ culprit(ellen)
¬culprit(frances)
==>? (answer 1 at end)

culprit(dolly) ∨ culprit(frances)
¬culprit(frances) ∨ culprit(harriet)
==>? (answer 2 at end)

Compare Resolution and Natural Deduction

c(d) ∨ c(f) + ¬c(f) ∨ c(h)
==> c(d) ∨ c(h) (by resolution)

(1) c(d) ∨ c(f) (Given)
(2) c(f) → c(h) (Given)

(3) c(d) (Ass)
(4) c(d)∨c(h) (∨I,3)

(5) c(f) (Ass)
(6) c(h) (→E,5)
(7) c(d)∨c(h) (∨I,6)

(8) c(d) ∨ c(h) (∨E, 3-4,5-7)

How do we obtain clausal form?

∀X(culprit(X) → inhouse(X))
¬culprit(X) ∨ inhouse(X)

culprit(X) and ¬culprit(X) are called *literals*.

P(..) ∨ Q(...) is called a *clause*

¬culprit(X) ∨ inhouse(X)
culprit(dolly)
==>? (answer 3 at end)

¬(¬culprit(dolly) ∧ ¬inhouse(dolly))
==>? (answer 4 at end)

Resolution and Factoring

are the two main steps uses in clausal theorem proving

and their definitions are given at the end on Slides 22/23

THE OTTER FAMILY OF PROVERS

Otter was developed at Argonne by Wos

- general purpose
- allows user control over lots of parameters
- allows reasoning with equals (=)

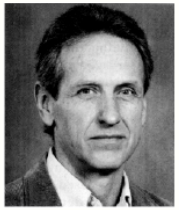
May be better ways for particular problems:

e.g. Davis Putnam for propositional logic
constraint solvers (eg for SODUKO)
PVS for program reasoning
temporal logic for model checking

Some Famous Names in Theorem Proving

Chang and Lee
First book on theorem
Proving (1968)

Alan Robinson
Discovered Resolution
1961



Robert Kowalski
Theory of Logic
Programming (1979)
Connection Graphs (1976)

Hillary Putnam
Davis Putnam
prover 1957

Alan Bundy
Clam Theorem
Prover,
excellent book,
problem solving

Larry Wos
Wrote Otter
prover
(1980s) and
successor
Prover9

You can find some more famous names in theorem proving at the end on Slides 30 and 31

and on slide 32 a diagram of various research areas of Computer Science that use theorem proving

An Example: Three naughty children

Dolly, Ellen or Frances was the culprit and only one.
The culprit was in the house.

Dolly: " It wasn't me, I wasn't in the house; Ellen did it."

Ellen : " It wasn't me and it wasn't Frances; but if I did do it then (Dolly did it too or was in the house)."

Frances: " I didn't do it; Dolly was in the house; if Dolly was in the house and did it so did Ellen."

None of the three told the truth.

Who did it?

c(d)∨c(e)∨c(f)	At least one of 3 girls did it
¬c(d)∨¬c(e) ¬c(d)∨¬c(f) ¬c(e)∨¬c(f)	Exactly one of them was the culprit
¬c(X)∨h(X)	The culprit was in the house
c(d)∨h(d)∨¬c(e)	Negation of dolly's testimony
c(e)∨c(f)∨c(e) c(e)∨c(f)∨¬c(d) c(e)∨c(f)∨¬h(d)	Negation of Ellen's testimony ≡ c(e)∨c(f) implied by above also implied by above
c(f)∨¬h(d)∨h(d) c(f)∨¬h(d)∨c(d) c(f)∨¬h(d)∨¬c(e)	Negation of Frances's testimony ≡ true
¬c(f)	Negation of goal (see Slide 24)

Proof found by Otter

See how to use Otter on Slide 25

```
1 [] c(d)|c(e)|c(f).
2 [] -c(d)| -c(e).
5 [] -c(X)|h(X).
6 [] c(d)|h(d)| -c(e).
7 [] c(e)|c(f)|c(e).
11 [] c(f)| -h(d)|c(d).
13 [] -c(f).
14 [binary,1.1,5.1,unit_del,13] c(e)|h(d).
      c(e)|c(f)|h(d) ==> c(e)|h(d) by 13
18 [binary,6.1,5.1,factor_simp] h(d)| -c(e).
      h(d)|h(d)|-c(e)==>h(d)|-c(e) by factor
20 [binary,18.2,14.1,factor_simp] h(d).
      h(d)|h(d)==>h(d) by factor
23 [binary,7.2,13.1,factor_simp] c(e).
      c(e)|c(e)==>c(e) by factor
25 [binary,23.1,2.2] -c(d).
26 [binary,11.1,13.1,unit_del,20,25] $F.
```


AUTOMATED REASONING IN OUR COURSE

Prolog language - year1
Artificial Intelligence - year 2
Software verification - year 3
Machine Learning - year 3

Automated Reasoning - year 4
Probabilistic Inference and Data Mining - year 4
Cognitive Robotics - year 4
Multi-Agent Systems - year 4

plus small bits here and there in other courses

Some more examples

and questions for you to try

are on Slides 26 -29

Project Ideas

1. Prolog viewed as a theorem Prover
   - Explain how Prolog operates as a theorem prover
   - What are its inference rules, data format, etc?
   - How does Prolog relate to Natural Deduction?
   - Find out how to use Prolog to write variants of itself!
2. Theorem proving with propositional logic
   - Investigate efficiencies when no variables present
   - Look at the Davis Putnam method
   - Look at the Model Generation method
   - Implement Davis Putnam (in Haskell or Java)
3. Theorem proving with Tableau
   - Investigate tableau proofs (KE and LeanCop)
   - Implement a tableau prover in Haskell (or Java)

Project Ideas Continued

4. Using Otter
   - How does Otter work? What are its inference rules?
   - Use Otter to solve the problems in these slides
   - What is the TPTP database? Write a testbed for testing Otter on problems from TPTP
5. Application
   - Find an application of automated theorem proving - eg program reasoning.
   - Find a prover that deals with the application and make up some problems for your application and explain the solutions; eg Perfect Developer or KEY system or PVS can be used for Program reasoning.

Even more Project Ideas

6. Theorem proving with Constraints
   - Find out how constraint solving can be included into reasoning. There are specialised constraint solvers that can be used for Soduko-like problems.
7. Theorem proving with Equality (=)
   - What are the extra rules to deal with Equality?
   - What help does Otter give for this?
   - What is rewriting and the Knuth Bendix procedure?
   - Implement a simple version of Knuth Bendix

SOME RESOURCES

These slides, a brief Introduction to Otter, and my AR notes: www.doc.ic.ac.uk/~kb/StudentsNew.html
Otter Manual and Website: www-unix.mcs.anl.gov/AR

Computer Modelling of Mathematical Reasoning by *Bundy*
Logic Programming by *Hogger*
Artificial Intelligence by *Russell and Norvig*
Computational Intelligence by *Poole et al*
Symbolic Logic and Mechanical Theorem Proving by *Chang and Lee*
Logic in Computer Science by *Huth and Ryan* (for model checking)

Perfect Developer (for reasoning about programs)

See also Slides 33/34

Resolution

Given: ¬P(X..) ∨ Q(...) ∨ R(...) and P(Y...) ∨ S(...)

To form the resolvent:
- match P(X....) and P(Y...)
  by finding values for variables in X... and Y....
  called unification – the result is unifier θ
- apply values to both clauses
- form resolvent = [Q(...) ∨ R(...) ∨ S(...)]θ

Example: ¬culprit(X) ∨ inhouse(X) and culprit(dolly)
Match: ¬culprit(X) and culprit(dolly)==> X/dolly (this is θ)
Resolvent: [inhouse(X)] θ ==> inhouse(dolly)

Factoring

Another useful operation is Factoring

In the simple case factoring merges identical literals

e.g P ∨ Q ∨ Q ==> P ∨ Q

More generally must first unify two literals (of same sign) with θ and then merge:

P(X)∨Q(X)∨Q(a) ==> P(a)∨Q(a)∨Q(a) (match X/a)
==> P(a)∨Q(a) (merge Q(a))

Why did we have to "know" the answer was c(f)?

Goal is to "find X such that c(X)" – i.e. ∃X.c(X)
Negate goal ==> ¬∃X.c(X) ≡ ∀X.¬c(X)

If add ¬c(X) to Otter it will find the solution
c(d)∨c(e)∨c(f) + ¬c(X1) ==> c(e)∨c(f)
c(e)∨c(f) + ¬c(X2) ==> c(f) and c(f) + ¬c(X3) ==> False
Otter uses 3 copies of ¬c(X),
each time with a fresh variable

Different goal (negated):
¬∃X,Y,Z [c(X) ∧ ¬c(Y) ∧ ¬c(Z) ∧ X≠Y ∧ X≠Z ∧ Y ≠Z]
≡ ¬c(X) ∨ c(Y) ∨ c(Z) ∨ X=Y ∨ X=Z ∨ Y =Z

How to use Otter

The data file has commands and data.
For enough information to get started,
see IntroOtter3 under topics notes in CATE

Example

```
set(binary_res).
set(prolog_style_variables).
set(interactive_given).
%a comment
%| means or, - means not
list(sos).
c(d)|c(e)|c(f).
-c(X)|h(X).
%<Etc.>
end_of_list.
```

When Otter runs it produces lots of statistics, such as number of resolvents, number of clauses generated, etc.

For this example, 43 clauses were generated.

Another Example - A harmonious household (!)

Someone who lived in the house stole from Aunt Agatha.
Agatha, the butler and James live in the house;
they are the only people that do.
A thief dislikes, and is never richer than, his victim.
James dislikes no-one whom Aunt Agatha dislikes.
Agatha dislikes everyone except the butler.
The butler dislikes anyone not richer than Agatha
and also everyone she dislikes.
No-one dislikes everyone.
Agatha is not the butler.
==> Agatha stole from herself
(and neither James nor the butler stole from her)

And one more still! - A Mathematical Problem

a $\circ$ b = c

$\circ$ is an associative operator:
x $\circ$ (y $\circ$ z) = (x $\circ$ y) $\circ$ z

x $\circ$ e = e $\circ$ x = x
e is the *identity* of $\circ$

x $\circ$ x = e

==> b $\circ$ a = c

EXERCISES

Check the translation into clausal form of 3 girls
Try the problem yourself

Translate the Harmonious Household data into logic
Try running it in Otter

Translate the mathematical problem into logic
(it almost is!)
Will Otter solve it for you?

Compare yourself with Otter on the Surprise (SI 29)

What else can Otter do?

A Simple but Surprising Problem

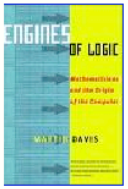
Try this one yourself and then in Otter
The size of the search is a big surprise

The relation p(x,y) is transitive
The relation q(x,y) is transitive and symmetric
Either p(x,y) or q(x,y) (for every x and y)
It is not the case that p(a,b)
It is not the case that q(c,d)

Draw a picture with four points a,b,c,d
Let p(x,y) mean a green line between x and y
Let q(x,y) mean a blue line between x and y
Show the given properties cannot all be true

Some Famous Names in Theorem Proving

Reiner Hahnle
LeanTap 1997 and other provers
Key System for Program Correctness 2005



Martin Davis
Davis Putnam Algorithm (1957)

Gerard Huet
Discovered rewrite systems 1976

Donald Knuth
Knuth Bendix Algorithm (1971)
Tex

J S Moore
Boyer MooreProver (1970s)
Structure Sharing (1971)

More Famous Names in Theorem Proving (No pictures!)

Donald Loveland - discovered Model Elimination 1967

Alain Colmerauer - first Prolog System

Mark Stickel - PTTP Theorem Prover 1982

David Plaistead - Hyper-linking (1992)

Reinhold Letz - Setheo, Free variable Tableau proving (1990s onwards)

Norbert Eisinger - Connection Graph theory (1986)

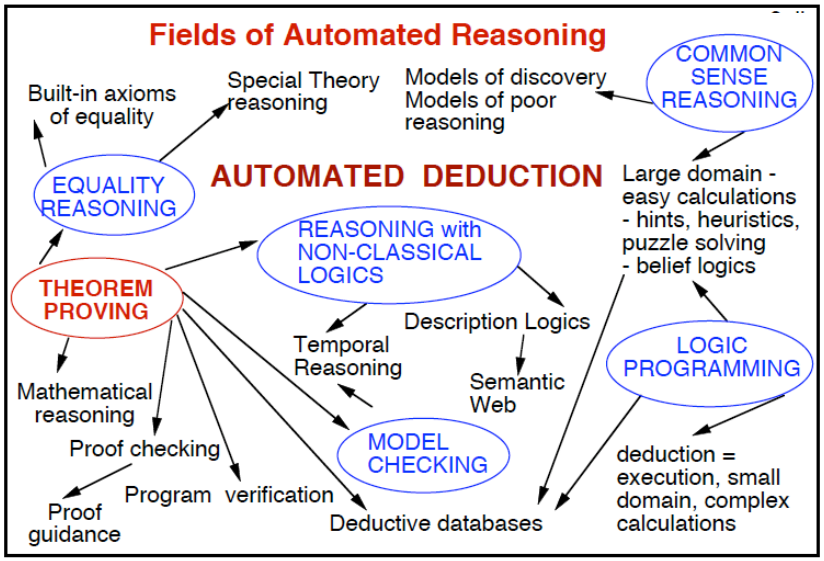
Woody Bledsoe - UT prover (natural deduction style)

Larry Paulson - Inventor of Isabelle

Andrei Voronkov - Vampire Theorem Prover

Ian Horrocks - Description Logic Theorem Prover

Fields of Automated Reasoning



Some Useful References and Websites

Alan Bundy: The Computer Modelling of Mathematical reasoning (Academic)

Chang & Lee: Symbolic Logic and Mechanical Theorem Proving (Academic)

Mel Fitting: First order Logic and Automated Theorem Proving (Springer)

Alan Robinson: Logic:Form and Function (Edinburgh)

Larry Wos: Automated Reasoning: Introduction and Applications (McGrawHill)

J. Kalman: Automated reasoning with Otter (Rinton)

Blasius & Burckert: Deduction Systems in Artificial Intelligence (Ellis Horwood)

Lassez & Plotkin (Eds): Computational Logic (MIT)

Kakas & Sadri (Eds): Computational Logic: Logic Prog and Beyond (Springer)

R. Veroff: Automated Reasoning and its Applications (MIT)

J. Gallier: Logic for Computer Science: Foundations of Automated Theorem Proving (Harper Row)

Several "freetech" books from www.freetech**books**.com

Useful References and Websites Continued

CADE: Conference on Automated Deduction (Springer)
http://www.cadeconference.org/ (old: http://www.cs.albany.edu/~nvm/cade.html)

TABLEAUX: Conference on Analytic Tableau and Related Methods (Springer)
http://i12www.ira.uka.de/TABLEAUX/

Workshop on First Order Theorem Proving http://www.csc.liv.ac.uk/FTP-WS/
(Old: http://www.mpi-sb.mpg.de/conferences/FTP-WS/)

http://en.wikipedia.org/wiki/Automated_theorem_proving
(Slightly biased contents)

Theorem Proving at Argonne
http://www-unix.mcs.anl.gov/AR/ (Includes Otter theorem Prover)

http://www.uni-koblenz.de/~beckert/leantap/ (A tableau based prover)

http://www.leancop.de/ (Another tableau based prover)

http://www.cl.cam.ac.uk/research/hvg/Isabelle/

http://alloy.mit.edu/

http://www.cs.miami.edu/~tptp/ (Thousands of problems for Theorem Provers)

http://www.cs.swan.ac.uk/~csetzer/logic-server/software.html (List of Provers)

Answers to questions

Q1:
culprit(dolly) v culprit(ellen) (by resolution)

Q2:
culprit(dolly) v culprit(harriet) (by resolution)

Q3:
inhouse(dolly) (by resolution)

Q4:
culprit(dolly) v inhouse(dolly)