Computational Logic

Introduction

Damiano Zanardini

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

Academic Year 2009/2010

Presentation

Goal

- get in touch with the *theoretical foundations* of Computational Logic
- (un)satisfiability as a criterion for deduction
- systematic *proof techniques*: know and apply
- study *resolution* with unification
- a glance at *Automated Theorem Proving* (ATP)
- towards *Logic Programming* (LP)

Formal Logic vs. Computational Logic

- in theory: FL goes before CL
- in practice: we are studying both in parallel, so we try to take this into account

Presentation

How to pass the exam...

- a written *test*
- probably, additional work during the course

☞ it is *not* enough to attend to classes: this is part of the basic contents of the master, so that you are supposed to really learn and be able to do something

Talking to me...

☞ you are *strongly invited* to come and ask me questions!

☞ you are *even more strongly invited* to interrupt me during classes

- any time (Mon→Fri), contact by mail; especially:
  - Tuesday 15.00 → 18.00 Thursday 10.00 → 13.00
- mail: `damiano@fi.upm.es` or `damiano.zanardini@gmail.com`

☞ please don't use gmail chat: write a mail instead

- my office: DIA, N. 2205 (tel. 913366901)

Contents

- first-order logic: syntax and semantics (recall, see FL course)
- standardization of formulæ
- Herbrand universe, Herbrand base, Herbrand interpretations
- automatic proof techniques
  - theory: semantic trees, Herbrand's theorem
  - ground methods: Gilmore, Davis-Putnam, Robinson's resolution
- resolution with unification
- resolution strategies
- extra (1): automated theorem proving
  - implementations
  - applications
- extra (2): towards logic programming
  - Horn clauses, SLD resolution, logic programs
  - applications

References

Books, lecture notes and papers

- C-L. Chang and R.C-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. 1973. [FI]
- J. Cuena. *Lógica Informática. Tomo II: Lógica Computacional*. 1999/2000. (in Spanish; something about program analysis) [FI]
- J.H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. 2003. (quite complete) [FI,PDF]
- M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. 2004. (mostly formal logic + program analysis) [FI,PDF]
- L. de Ledesma. *Lógica para la computación*. 2009. (in Spanish) [FI]
- A. Leitsch. *The Resolution Calculus*. 1997. [E.U. Informática]
- L. Paulson. *Logic and Proof*. 2007. (shorter) [PDF]

References

Books, lecture notes and papers

- A. Ramsay. *Formal Methods in Artificial Intelligence*. 1989 [FI]
- J.A. Robinson. *Computational Logic - Memories of the Past and Challenges for the Future*. 2000. (interesting paper)
- R.M. Smullyan. *First-order logic*. 1995 (orig. 1968) [FI]
- T. Tymoczko and J. Henle. *Sweet Reason*. 1995. (a broader view on logic; definitely good for a read) [FI]
- Many Authors from UPC. *Notas de Clase para IL*. 2009. in Spanish, but quite interesting. [I have a copy]

Links and Resources

On the WWW

The old webpage (2008/2009)

- `http://clip.dia.fi.upm.es/~damiano/teaching/emcl/cl_08_09/`

The new one

- `http://web3.fi.upm.es/AulaVirtual/course/view.php?id=141`
- password for guest access: lcomp

I will put here slides, exercises and any useful stuff about the course

The *core business* of Computational Logic

The *formal* side

- representing facts in a formal language
- defining techniques for *deducing* new facts (theorems)

The *computational* side

- the same formal machinery, more or less
- this time, the deduction is *automatic* (computable)

The *core business* of Computational Logic

Mind...

- ...the difference between proofs as *discoveries* and proofs as *computations*
- ...proof generation vs. proof checking
- ...the difference between Deep Blue and Garry Kasparov
- ...the (frustrated) goal of David Hilbert and the *formalists*
- ...the happiness you would feel if being told *every thing you want to do, I have an automatic tool which gives you the good program for it*!

We could say that there is a tradeoff between

- *power*: how difficult is what we can do
- *simplicity*: how easily (and automatically) we can do things

The *core business* of Computational Logic

A machine can be *stupid enough* to...

- ...make a useless step
  - Deep Blue: not pruning the tree
  - Apply all the possible legal rules to the current state, which usually amounts to a lot of useless work
- ...make the least-efficient step
  - In resolution, resolve between two big clauses in the first step
  - Prove $F \wedge F$ without noticing that we can prove F only once
  - (more subtle) Proving $F \rightarrow G$ starting from proving F
- ...make one step forward, one backward, one forward, ...
  - $F \rightsquigarrow F \wedge F \rightsquigarrow F \rightsquigarrow F \wedge F \rightsquigarrow F$...
- ...always make the first step it finds among the legal ones (more tricky)
  - $F \rightsquigarrow F \wedge F \rightsquigarrow (F \wedge F) \wedge (F \wedge F)$...
  - Naïf implementation of simple recursive definitions: (1) Checking $\subseteq$; (2) Prolog predicate for Java subclasses
  - Robinson's example about non-terminating resolution: $\{p(a), \neg p(x) \vee p(f(x))\}$
  - In general, non-termination of depth-first algorithms