SFU
SIMON FRASER UNIVERSITY
THINKING OF THE WORLD

UNIVERSITAT ROVIRA I VIRGILI

Teaching Logic Programming Tools for Interdisciplinary Computing

G. Bel Enguix, M.D. Jimenez Lopez, Veronica Dahl
Chair of Excellence, E.C.
Professor and Lab Director, SFU

Abstract

Teaching logic programming as the axis for *interdisciplinary research around AI*.

In particular, which approaches to teaching it can appeal to people from the humanities, and for women, notably underrepresented in computing sciences.

Motivation

- - interdisciplinarity is becoming ubiquituous
- - researchers outside C.S. are in dire need of AI tools
- - languages that are closer to human communication modes (e.g. inferential capabilities) can help bridge the gap more readily than traditional languages
- - those transmuting more readily into linguistic tools are likely to appeal to women, typically underrepresented in C.S.

Motivation- socioeconomic

- - the demand for ICT skilled professionals is predicted to reach 250,000 by 2010, whereas only 180,000 are likely to be available (assessment by the e-skills Industry Leadership Board, Jan. 2009)

LP as a Promising Paradigm for Interdisciplinary Research

Twofold appeal:

- - its "instructions" take human-like reasoning forms
- - different components of the same system can be expressed in Prolog and interact with each other easily because of this uniformity of expression

But how promising to non-computer experts?

- - one must distinguish two levels

Some Relevant LP Features

- Procedural vs. Declarative Interpretations.
- member(X,[X|_]).
  member(X,[_|L]):- member(X,L).

First line reads: "*X is a member of any list whose head is X*", (declarative) and also:

"*to prove that X is a member of a given list, prove that it is the head of that list*" (proc.)

Procedural /Declarative Gap

No longer coincide if extra or metalogical features, such as “cut”, are used. E.g.

member(X,[X|_]):- !.

To be addressed early on using simple examples (for instance on the uses of cut, or on loop avoidance)

Modularity and Metaprogramming for Rapid Prototyping

- - inherent modularity: useful for testing different theories
- - we can also experiment with the proof process itself => metaprogramming

Linguistic Friendliness and Affinity

- - Natural formats:
  - - *assertion* format- which can be quite general by its ability to contain (universally quantified) variables
  - - *rule* format, stating under which conditions a given fact to be proved holds.
- - Mnemonic names for constants, function names and predicates.

Linguistic Friendliness and Affinity

- - Through grammatical instances, better able to reach general audiences oriented towards the humanities, and in particular, better able to reach female audiences.

Where to Start- Logic Theory versus Plunging in

- first order logic and automated theorem proving?

- - theoretical apparatus often intimidating
- - logic has a reputation for being “difficult”

Therefore: teach some of its concepts as a by-product of using it to program

Where to Start- pure Prolog or its grammatical version?

- For linguistics students, as well as for students having learned formal languages and/or compiler theory, it makes sense to start with logic grammars.
- For some reason, unification is immediate for these students to understand in the case of natural language, but tricky in the case of other applications.
- Also the case for other students in the humanities

Translation Systems

-usually interesting to students of literature, linguistics, philology.

translate([u,u,a],leucine).

translate([g,c,a],alanine).

s([u,u,a]) → [leucine].

s([g,c,a]) → [alanine].

Evaluating Results- Philology Assisted by logic programming

- the non-traditional approach of plunging into grammars first and Prolog next seems to work better for mixed audiences;
- translation systems should be taught earlier than more general information systems;
- capacity for abstract thinking is often more important than the ``right'' computer programming background;
- symbolic processing easier than number crunching.

Evaluating Results

practical concepts should precede theoretical concepts;

focusing at first on programming elegance and conciseness yields better results than worrying too much about efficiency;

Surprisingly, almost all students judged that the programming skills learnt were going to be useful in their future studies and work, although most of them were majoring in humanistic sciences.

Conclusions

many students could benefit from exposure to high level computing even if never destined to program by themselves;

in particular, the pool of female students now absent from computing sciences careers and related interdisciplinary endeavors, would be perhaps more easily included through the starting point of logic grammars and logic programming.

Thanks!