Logic Programming in a Fragment of Intuitionistic Linear Logic

by

Joshua Hodas, PhD student at UPenn
[now an attorney in Los Angeles]

and

Dale Miller, Edinburgh (sabbatical leave from UPenn)
[now INRIA]

A revised version appears in Information and Computation, 1994.

Remembering the early 1990's

Two new, exciting innovations:

- linear logic [1987]
- π-calculus [1989]

Many areas of computational logic, concurrency theory, and programming language semantics have been influence by them.

. . . but there was a steep learning curve.

Linear logic was strange: proof nets, slices, phase semantics, additive/multiplicative/exponential connectives, etc.

The LICS 91 paper showed that

- logic programming became more expressive using linear logic, and
- linear logic programming had applications.

In the '91 and '94 papers

$\mathsf{Lolli} = \{\top, \&, \forall, \supset\} \cup \{\multimap\}$

- Linear logic without exponentials: $\mathsf{LL} = \mathsf{Lolli} \cup \{\bot\}$
- Completeness of "goal directed search"
- A polarized embedding of intuitionistic logic into linear logic (needs half as many exponentials).
- A canonical model given as a resource indexed Kripke model
- Lazy splitting of contexts
- Several applications.

(from LICS91) Aspects of Intuitionistic Contexts

Theorem Proving

+ Contexts manage hypotheses and eigen-variables elegantly.
− Contraction cannot be controlled naturally.

Linguistics

+ Relative clauses are sentences with noun phrase gaps: $(NP \supset SENT) \supset REL$.
− Gap extraction is non-vacuous and satisfy island constraints

Data Bases

+ Contexts can act as databases and support query answering by deduction.
− Contexts cannot naturally be "edited" or updated.

Object State

+ Objects can have their state and methods hidden in a context.
− Updating object state is not possible declaratively.

The linear logic extension changed the minuses to pluses.

A word about the future (paraphrasing *The Graduate*)

Mr. McGuire: I just want to say one word to you. Just one word.
Ben: Yes, sir.
Mr. McGuire: Are you listening?
Ben: Yes, I am.
Mr. McGuire: *Focused proof systems*
Ben: But isn't that three words?

Focused proof systems provide control of the structural rules without a direct appeal to linear logic. They provide remarkably flexible normal forms.

Completeness of a focusing proof systems is the *second* most important result about a sequent proof system for CS applications.

Mr. McGuire: But what is the most important result?
Ben: Cut-elimination, of course.

— Thank you —