Forward and Backward Chaining in Linear Logic

James Harland[1] David Pym[2] Michael Winikoff[1]
jah@cs.rmit.edu.au, pym@dcs.qmw.ac.uk, winikoff@cs.rmit.edu.au

[1] Department of Computer Science, Royal Melbourne Institute of Technology, GPO Box 2476V, Melbourne, 3001, Australia
[2] Department of Computer Science, Queen Mary and Westfield College, University of London, Mile End Road, London, E1 4NS UK

Backward chaining is a standard technique in automated deduction, particularly in *logic programming* systems, often taking the form of a version of Robinson's *resolution rule* [10]. The fundamental question is to determine whether or not a given formula follows from a given set of formulæ, and there are various techniques which can be used to guide the search for a proof.

An instance of this approach is the analysis of logic programming in intuitionisitic logic [8]. The standard such analysis is that of Miller et al. [8], based around the notion of *uniform proofs*. These are defined in terms of the sequent calculus, which is well-known to be suited for analyses of backward-chaining.

With more recent interest in logic programming languages based on *linear logic* [5], the natural extension of these backward chaining techniques to linear logic has been much studied [1, 2, 4, 6, 9]; generally, it follows a similar pattern to intuitionistic logic. The details of the analysis are more intricate than in intuitionistic logic, and there are a number of points of diversion amongst the various approaches, but the same general procedure is followed.

Whilst the sequent calculus is a good basis for backward chaining, other systems for inference in intuitionistic logic provide forward chaining capabilities. Hilbert-type systems are the oldest and perhaps best-known of such systems [7]. Such systems allow different logics to be specified by different sets of axioms whilst maintaining modus ponens as the sole means of inference.

Another technical expression of forward chaining in intuitionistic logic may be found in the T_P operator used in the semantics of logic programs. Here, a mapping is made from interpretations to interpretations, in which the image is the result of applying the rules of the program to the initial interpretation via a combination of modus ponens and unification. The semantics of the program is then given by the least fixed point of this operator. It is interesting to note that this forward chaining system is traditionally used to provide a fixpoint semantics for SLD-resolution [3], a backward-chaining system.

It should be noted that a key property of the modus ponens rule in intuitionistic logic is that it preserves equivalence: $\phi \wedge (\phi \supset \psi) \equiv \phi \wedge \psi$. This strong property greatly simplifies the analysis of this rule of inference.

A combination of both backward and forward chaining may be found in deductive database systems such as Aditi [11]. In such systems, which are based on variants of Prolog, forward chaining is generally used in order to compute all answers to a query using efficient join algorithms and other techniques from relational databases, whilst backward chaining is used for less data-intensive computational tasks (such as format conversions).

The properties of forward and backward chaining systems for intuitionistic logic are generally well-understood. However, the question of how to best integrate the two models still remains. Give that Hilbert-type systems generally do not make any provision for backward chaining techniques, it seems reasonable to address the question of integration by investigating the incorporation of forward chaining features into the sequent calculus.

This is achieved by inserting *directed cuts* into an otherwise cut-free sequent calculus proof. Cut-free proofs are generally used in proof search to avoid a significant amount of non-determinism (*i.e.*, having to choose a cut formula arbitrarily); in the case of a directed cut, the cut formula will be calculated from the antecedent, and hence will not have the same problem.

The presence of both backward and forward chaining mechanisms in deductive databases suggests that a similar integration for linear logic will prove fruitful, especially as linear logic has been used to model database updates, state and action problems and concurrency.

However, the use of modus ponens in linear logic is not as simple as in intuitionistic logic since modus ponens does not preserve linear equivalence. For example, in linear logic $p \otimes (p \multimap q) \vdash q$ but in proving q we have to "consume" p.

We have recently developed a method for integrating forward chaining into the sequent calculi for intuitionistic and linear logics. One particular result of interest is that we can show that these inference rules respect an encoding of intuitionistic logic programs into linear ones.

Acknowledgements Harland is grateful for the hospitality of the Department of Computer Science of Queen Mary and Westfield College, University of London during a period of sabbatical leave. Pym is partially supported by a Fellowship from the EPSRC.

References

1. J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *J. Logic Computat.* 2(3), 1992.
2. J.-M. Andreoli and R. Pareschi. Linear Objects: Logical Processes with Built-in Inheritance. Proceedings of the International Conference on Logic Programming, 496-510, Jerusalem, June, 1990.
3. M.H. van Emden and R.A. Kowalski, The Semantics of Predicate Logic as a Programming Language, *Journal of the Association for Computing Machinery* 23:4:733-742, October, 1976.
4. D. Galmiche and G. Perrier. On proof normalization in Linear Logic. Theoretical Computer Science 135:76-100, 1994.
5. J.-Y. Girard. Linear Logic. *Theoretical Computer Science* 50, 1-102, 1987.
6. J. Hodas, D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic: Extended Abstract. Proceedings of the Symposium on Logic in Computer Science, 32-42, Amsterdam, July, 1991.
7. S.C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.
8. D. Miller, G. Nadathur, F. Pfenning and A. Ščedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic:51:125-157*, 1991.
9. D.J. Pym, J.A. Harland. A Uniform Proof-theoretic Investigation of Linear Logic Programming. Journal of Logic and Computation 4:2:175-207, April, 1994.
10. J.A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the Association for Computing Machinery* 12:1:23-41, 1965.
11. J. Vaghani, K. Ramamohanarao, D. Kemp, Z. Somogyi, P. Stuckey, T. Leask and J. Harland. The Aditi Deductive Database System. *VLDB Journal* 3:2:245-288, April, 1994.