A Survey of Linear Logic Programming

Dale Miller

Computer Science Department
University of Pennsylvania
Philadelphia, PA 19106-6389 USA
dale@saul.cis.upenn.edu

A hypertext version of this survey can be found at
ftp://www.cis.upenn.edu/pub/papers/miller/ComputNet95/llsurvey.html.
This survey will appear in the third issue of the
Newsletter of the Network of Excellence on Computational Logic.

1 Introduction

It is now common place to recognize the important role of logic in the foundations of computer science in general and programming languages more specifically. For this reason, when a major new advance is made in our understanding of logic, we can expect to see that advance ripple into other areas of computer science. Such rippling has been observed during the past eight years since the first introduction of linear logic [Girard 1987]. This exciting advance in logic provides new ways of embracing aspects of computation directly in a rich logical framework. Since this development extends and enriches our understanding of classical and intuitionistic logic, it provides new insights into the many computational systems built on those two logics.

2 Applications of Linear Logic to Programming Languages

Both functional and logic programming have made extensive use of classical and intuitionistic logic to motivate language designs and to analyze programs and specifications.

One inspiration for the design of functional programming languages is the Curry-Howard Isomorphism. This isomorphism states that programs and proofs can be equated and that the normalization of proofs (say, by beta-conversion or cut-elimination) can be seen as computation. Linear logic supplies new proof structures, called proof nets, and the dynamics of their normalization can be used to express some aspects of concurrency [Abramsky 1993, Bellin & Scott 1992, Lafont 1989, Lafont 1990]. The Curry-Howard Isomorphism also states that the types of programs can be seen as formulas, and the richer formulas of linear logic allow for more expressive types. Such stronger types have been used to help provide static analysis of such things as run-time garbage, aliases, reference counters, and single-threadedness [Guzmán & Hudak 1990, O'Hearn

1991, Maraist et. al. 95, Wadler 90, Chirimar et.al.].

Linear Logic has also shown promise in helping with the analysis of conventional logic programs. See, for example, the work of Cerrito on specifying the semantics of various aspects of Prolog using linear logic [Cerrito 1990, Cerrito 1992b, Cerrito 1992a] and of Reddy in specifying modes using linear logic [Reddy 1993]. The most active work on using linear logic in logic programming, however, has been in the area of designing and using new logic programming languages.

3 New Logic Programming Languages

In the field of logic programming there does not seem to be a principle, like that of the Curry Howard Isomorphism, that is at once simple, natural, deep, and generally accepted as a design principle. Although most early work on logic programming had been fixed on a particular logic, namely that of first-order, classical Horn clauses, many researchers have recently adopted proof search in sequent calculi as a setting for designing and exploring the dynamics and properties of logic programs. In this setting, sequents are used to denote the state of a computation and the transformations that occur to sequents as cut-free proofs are incrementally constructed are used to model the dynamics of computation.

When few logical connectives are needed, it is often straightforward to define a logic that has a natural operational semantics (meaning that it is easy for a programmer to understand how proofs are attempted). The following three designs are examples of such linear logic programming languages.

- LO (Linear Objects) [Andreoli & Pareschi 1990, Andreoli & Pareschi 1991] was designed by Andreoli and Pareschi as an extension to the Horn clause paradigm in which atomic formulas are generalized to be multisets of atomic formulas connected by multiplicative disjunctions ("pars"). In LO, backchaining becomes multiset rewriting. This language has been used to specify object-oriented programming and the coordination of processes.

- ACL by Kobayashi and Yonezawa is an asynchronous calculus in which the send and read primitives were essentially identified to two complementary linear logic connectives [Kobayashi & Yonezawa 1993, Kobayashi & Yonezawa 1994].

- Lincoln and Saraswat, in unpublished reports, developed a linear version of concurrent constraint programming and used linear logic connectives to extend previous languages in this paradigm [Lincoln & Saraswat 1993, Saraswat 1993].

Each of these languages incorporate small subsets of linear logic and use multisets of formulas to capture object structure or collections of processes and messages, and use multiset rewriting to capture inheritance and synchronization.

One design principle that has been used in recent years states that *goal-directed search* should be complete for logic programs. Within intuitionistic logic, this was first formalized using the the proof theoretic notion of *uniform proofs* [Miller et.al 1991]. Horn clause logic and hereditary Harrop formulas (the logic underlying λProlog) are both examples of settings where goal-directed search is complete for intuitionistic provability. This definition of goal-directed search depends on the fact that sequents in intuitionistic logic are *single-conclusion*; that is, they contain a single conclusion (the goal) to be proved. It was therefore straightforward to extend the definition of uniform proofs to intuitionistic linear logic (where sequents again have a single conclusion), and Hodas and Miller have used that extension to design the linear logic programming language Lolli [Hodas 1994, Hodas & Miller 1994]. Lolli can be seen as a modular extension to λProlog that allows items in the program to be use either once or an unlimited number of times. Linear logic's connectives can be used to provide elegant and flexible management of both kinds of program clauses.

In the sequent characterization of full linear logic, sequents can have multiple formulas in the conclusion: that is, the goal to be proved may be a multiset of formulas whose provability cannot be isolated from each other and where each formula can assist each other in some (hopefully, programmable) sense. Given this structure of goals, the notion of goal directed search and uniform proof needed to be extended. There appear to be two ways to make this extension. In one approach a goal with multiple parts is required to have *some* component goal that can be reduced. This approach, used by Harland and Pym, is the weaker approach and goal-directed search would be complete for many subsets of linear logic [Pym & Harland 1994], some of which have complex operational semantics [Harland & Pym 1992]. See their Lygon language [Harland & Winikoff 1995], for example. Another approach requires that in a goal with multiple component goals, *all* components must be simultaneously reducible. Miller first used this definition in [Miller 1992] to provide a linear logic encoding of the pi-calculus. He later showed that by selecting a suitable and complete set of connectives, all of linear logic can be seen as logic programming. This particular presentation of linear logic, called Forum [Miller 1994], can be seen (and motivated) as an extension of LO, λProlog, and Lolli (but not of Lygon). Hodas is currently developing a prototype implementation of Forum based on techniques used in the implementation of Lolli.

4 Applications for new languages

Not all of these designs have been undertaken as purely technical exercises involving the structure of sequent calculus proofs. In fact, mnay of these designs have been motivated by the need to make logic specifications more expressive. Some of the resulting systems have supplied new and useful specifications in various domains.

Concurrency Many of these programming languages were designed, at least in part, to allow concurrent specifications [Kobayashi & Yonezawa 1993, Kobayashi & Yonezawa 1994, Lincoln & Saraswat 1993, Miller 1992, Saraswat 1993]. See also [Bruscoli & Guglielmi 1995].

Object-oriented programming Capturing state and inheritance was an early goal of the LO system [Andreoli & Pareschi 1991] and a motivation for the design of Lolli. Another approach to state encapsulation can be found in [Miller 1994] and in [Delzanno & Martelli 1995].

Operational semantics Forum has been successfully used to specify the operational semantics of imperative and concurrent features such as those in Algol and ML [Chirimar 1995, Miller 1994]. Chirimar has also specified in Forum the operational semantics of a pipe-lined, RISC processor [Chirimar 1995].

Natural language parsing Lolli has provided a declarative approach to gap threading within English relative clauses [Hodas 1992].

Object-logic proof systems Lolli has been used to refine the usual, intuitionistic specifications of object-level natural deduction systems [Hodas & Miller 1994] and Forum has been used to provide specifications of object-level sequent systems [Miller 1994].

5 Research in Sequent Calculus Proof Search

Since the majority of linear logic programming are described using sequent calculus proof systems, a great deal of work in understanding and implementing these languages has focused on properties of proofs, rather than on model theoretic considerations. In recent years, results in proof theory have been developed specifically to support this foundation of the logic programming paradigm. Andreoli developed some deep results about proof search in linear logic in his PhD thesis [Andreoli 1990a] (see also [Andreoli 1990b]). There is also related work by Galmiche, Boudinet, and Perrier [Galmiche & Boudinet 1994, Galmiche & Perrier 1994], Tammet [Tammet 1994], and others. A problem specific to proof search in linear logic is how to effectively split resources between conjunctive branches of a computation. For Lolli, Hodas and Miller developed a lazy splitting approach, called the input-output model of resource consumption [Hodas & Miller 1994, Hodas 1994]: various researchers are actively refining and extending that model (see, for example, [Cervesato et. al.]).

Possible future projects include exploring how to exploit Girard's LU proof system [Girard 1993] and definitional reflection [Schroeder-Heister 1993]. Also, since linear logic is a rich and expressive logic, finding interesting subsets of it that can be given effective implementations is currently an open problem.

6 References

This list of references was compiled, in part, from the Bibliography on Linear Logic, maintained by I. Cervesato, F. Pfenning, and C. Schürmann.

[Abramsky 1993] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3-57, 1993.

[Andreoli 1990a] J.-M. Andreoli. *Proposal for a Synthesis of Logic and Object-Oriented Programming Paradigms.* PhD thesis, University of Paris VI, 1990.

[Andreoli 1990b] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297-347, 1992.

[Andreoli & Pareschi 1990] J.-M. Andreoli and R. Pareschi. LO and behold! Concurrent structured processes. In *Proceedings of OOPSLA'90*, pages 44-56, Ottawa, Canada, October 1990. Published as ACM SIGPLAN Notices, vol.25, no.10.

[Andreoli & Pareschi 1991] J.-M. Andreoli and R. Pareschi. Linear objects: Logical processes with built-in inheritance. *New Generation Computing*, 9:445-473, 1991.

[Bellin & Scott 1992] G. Bellin and P. J. Scott. On the pi-calculus and linear logic. Manuscript, 1992.

[Bruscoli & Guglielmi 1995] P. Bruscoli and A. Guglielmi. A Linear Logic Programming Language with Parallel and Sequential Conjunction, *Proceedings of the 1995 GULP-PRODE Joint Conference on Declarative Programming*, Marina di Vietri, Italy, 11-14 September 1995.

[Cerrito 1990] S. Cerrito. A linear semantics for allowed logic programs. In *Proceedings of the Fifth Symposium on Logic in Computer Science*, pages 219-227, Philadelphia, Pennsylvania, June 1990. IEEE Computer Society Press.

[Cerrito 1992b] S. Cerrito. A linear axiomatization of negation as failure. *Journal of Logic Programming*, 12:1-24, 1992.

[Cerrito 1992a] S. Cerrito. Negation and linear completion. In L. Farinas del Cerro and M. Penttonen, editors, *Intensional Logic for Programming*, pages 155-194. Clarendon Press, 1992.

[Cervesato et. al.] I. Cervesato, J. S. Hodas, and F. Pfenning, *Efficient Resource Management for Linear Logic Proof Search*, submitted to Extensions of Logic Programming 1996.

[Chirimar 1995] J. Chirimar. *Proof Theoretic Approach to Specification Languages.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.

[**Chirimar et.al. 1992**] J. Chirimar, C. A. Gunter, and J. G. Riecke. Proving memory management invariants for a language based on linear logic. *Lisp and Functional Programming*, pages 139-150, 1992.

[**Chirimar et. al.**] J. Chirimar, C. A. Gunter, and J. G. Riecke. Reference counting as a computational interpretation of linear logic. *Journal of Functional Programming*, to appear.

[**Delzanno & Martelli 1995**] G. Delzanno and M. Martelli. Objects in Forum. *Proceedings of the 1995 International Symposium on Logic Programming*, 4-7 December 1995, Portland Oregon.

[**Galmiche & Boudinet 1994**] D. Galmiche and E. Boudinet. Proof search for programming in intuitionistic linear logic. In D. Galmiche and L. Wallen, editors, *CADE-12 Workshop on Proof Search in Type-Theoretic Languages*, pages 24-30, Nancy, France, June 1994.

[**Galmiche & Perrier 1994**] D. Galmiche and G. Perrier. Foundations of proof search strategies design in linear logic. In *Symposium on Logical Foundations of Computer Science*, pages 101-113, St. Petersburg, Russia, 1994. Springer-Verlag LNCS 813. Also available as Technical Report CRIN 94-R-112 from the Centre di Recherche en Informatique de Nancy.

[**Girard 1987**] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1-102, 1987.

[**Girard 1993**] J.-Y. Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59:201-217, 1993.

[**Guzmán & Hudak 1990**] J. C. Guzmán and P. Hudak. Single-threaded polymorphic lambda calculus. In *Proceedings of the Fifth Symposium on Logic in Computer Science*, pages 333-343, Philadelphia, Pennsylvania, June 1990. IEEE Computer Society Press.

[**Harland & Pym 1992**] J. Harland and D. Pym. On resolution in fragments of classical linear logic. In *Proceedings of the Russian Conference on Logic Programming and Automated Reasoning*, pages 30-41. Springer-Verlag LNAI 624, July 1992.

[**Harland & Winikoff 1995**] J. Harland and M. Winikoff. Implementation and development issues for the linear logic programming language Lygon. In *Proceedings of the Eighteenth Australian Computer Science Conference*, pages 563-572, Adelaide, Australia, February 1995. Also available as Technical Report TR 95/6, Melbourne University, Department of Computer Science.

[**Hodas 1992**] J. S. Hodas. Specifying filler-gap dependency parsers in a linear-logic programming language. In K. Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 622-636, Washington, DC, November 1992.

[**Hodas 1994**] J. S. Hodas. *Logic Programming in Intuitionistic Linear Logic: Theory, Design and Implementation.* PhD thesis, University of Pennsylvania, Department of Computer and Information Science, 1994.

[**Hodas & Miller 1994**] J. S. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327-365, 1994. Extended abstraction in the Proceedings of the Sixth Annual Symposium on Logic in Computer Science, Amsterdam, July 15-18, 1991.

[Kobayashi & Yonezawa 1993] N. Kobayashi and A. Yonezawa. ACL - a concurrent linear logic programming paradigm. In D. Miller, editor, *Proceedings of the International Logic Programming Symposium*, pages 279-294, Vancouver, Canada, October 1993. MIT Press.

[Kobayashi & Yonezawa 1994] N. Kobayashi and A. Yonezawa. Asynchronous communication model based on linear logic. *Formal Aspects of Computing*, 3:279-294, 1994. Short version appeared in Joint International Conference and Symposium on Logic Programming, Washington, DC, November 1992, Workshop on Linear Logic and Logic Programming, edited by D. Miller.

[Lafont 1989] Y. Lafont. Functional programming and linear logic. Lecture notes for the Summer School on Functional Programming and Constructive Logic, Glasgow, United Kingdom, 1989.

[Lafont 1990] Y. Lafont. Interaction nets. In *Seventeenth Annual Symposium on Principles of Programming Languages*, pages 95-108, San Francisco, California, 1990. ACM Press.

[Lincoln & Saraswat 1993] P. Lincoln and V. Saraswat. Higher-order, linear, concurrent constraint programming. Manuscript, January 1993.

[Maraist et. al. 95] J. Maraist, M. Odersky, D. Turner, and P. Wadler. *Call-by-name, call-by-value, call-by-need, and the linear lambda calculus*, 11'th International Conference on the Mathematical Foundations of Programming Semantics, New Orleans, Lousiana, April 1995.

[Miller et. al. 1991] D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied logic*, 51:125-157, 1991.

[Miller 1992] D. Miller. The pi-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Proceedings of the Workshop on Extensions of Logic Programming*, pages 242-265. Springer-Verlag LNCS 660, 1992.

[Miller 1994] D. Miller. A multiple-conclusion meta-logic. In S. Abramsky, editor, *Ninth Annual Symposium on Logic in Computer Science*, pages 272-281, Paris, France, July 1994. IEEE Computer Society Press.

[O'Hearn 1991] P. W. O'Hearn. Linear logic and interference control: Preliminary report. In S. Abramsky, P.-L. Curien, A. M. Pitts D. H. Pitt, , A. Poigné, and D. E. Rydeheard, editors, *Proceedings of the Conference on Category Theory and Computer Science*, pages 74-93, Paris, France, 1991. Springer-Verlag LNCS 530.

[Pym & Harland 1994] J. Harland and D. Pym. A uniform proof-theoretic investigation of linear logic programming. *Journal of Logic and Computation*, 4(2):175-207, April 1994.

[Reddy 1993] U. S. Reddy. A typed foundation for directional logic programming. In E. Lamma and P. Mello, editors, *Third International Workshop on Extensions of logic programming*, pages 150-167, Bologna, Italy, 1993.

Springer-Verlag LNAI 660.

[Saraswat 1993] V. Saraswat. A brief introduction to linear concurrent constraint programming. Manuscript, 1993.

[Schroeder-Heister 1993] P. Schroeder-Heister. Rules of Definitional Reflection. In M. Vardi editor, *Eighth Annual Symposium on Logic in Computer Science*, pages 222-232, IEEE, June 1993.

[Tammet 1994] T. Tammet. Proof strategies in linear logic. *Journal of Automated Reasoning*, 12:273-304, 1994.

[Wadler 90] P. Wadler. Linear types can change the world. In M. Broy and C. B. Jones, editors, *IFIP TC 2 Working Conference on Programming Concepts and Methods*, pages 561-581, Sea of Gallilee, Israel, April 1990. North-Holland.

Acknowledgments

I wish to thank I. Cervesato, F. Pfenning, and C. Schürmann for maintaining the extensive Bibliography on Linear Logic: it proved valuable in assembling this survey. Josh Hodas and James Harland provided some useful comments on a draft of this document. I am also pleased to acknowledge support from ONR N00014-93-1-1324, NSF CCR-91-02753, NSF CCR-92-09224, and DARPA N00014-85-K-0018.