Linear Logic as Logic Programming: An Abstract*

Dale Miller

Computer Science Department, University of Pennsylvania
Philadelphia, PA 19104-6389 USA
dale@cis.upenn.edu
http://www.cis.upenn.edu/~dale

The theory of cut-free sequent proofs has been used to motivate and justify the design of a number of logic programming languages. Two such languages, λProlog and its linear logic refinement, Lolli [13], provide for various forms of abstraction (modules, abstract data types, and higher-order programming) but lack primitives for concurrency. The logic programming language, LO (Linear Objects) [2] provides some primitives for concurrency but lacks abstraction mechanisms. Forum is a logic programming presentation of all of linear logic that modularly extends λProlog, Lolli, and LO. This language, therefore, allows specifications to incorporate both abstractions and concurrency.

The motivation for Forum. Below are several examples of logic programming languages. Here we use linear logic connectives as in [9], with the addition of $\Rightarrow$ for intuitionistic implication: $A \Rightarrow B$ denotes $!A \multimap B$.

1. *Horn clauses*, the logical foundation of Prolog, are formulas of the form $\forall \bar{x}(G \Rightarrow A)$ where G may contain occurrences of & and $\top$. (We shall use $\bar{x}$ as a syntactic variable ranging over a list of variables and A as a syntactic variables ranging over atomic formulas.) In such formulas, occurrences of $\Rightarrow$ and $\forall$ are restricted so that they do not occur to the left of the implication $\Rightarrow$. As a result of this restriction, cut-free proofs involving Horn clauses do not contain right-introduction rules for $\Rightarrow$ and $\forall$.
2. *Hereditary Harrop formulas* [17], the logical foundation of λProlog, result from removing the restriction on $\Rightarrow$ and $\forall$ in Horn clauses: that is, such formulas can be built freely from $\top$, &, $\Rightarrow$, and $\forall$. Some presentations of hereditary Harrop formulas and Horn clauses allow certain occurrences of disjunctions ($\oplus$) and existential quantifiers [17]: since such occurrences do not add much to the expressiveness of these languages, they are not considered directly here.
3. The logic at the foundation of *Lolli* is the result of adding $\multimap$ to the connectives present in hereditary Harrop formulas: that is, Lolli programs are freely built from $\top$, &, $\multimap$, $\Rightarrow$, and $\forall$. As with hereditary Harrop formulas, it is possible to also allow certain occurrences of $\oplus$ and $\exists$, as well as the tensor $\otimes$ and the modal !.

* Parts of this abstract are taken from the paper [16].

4. The formulas used in LO are of the form $\forall \bar{x}(G \multimap A_1 \parr \cdots \parr A_n)$ where $n \geq 1$ and G may contain occurrences of $\&$, $\top$, $\parr$, $\bot$. Similar to the Horn clause case, occurrences of $\multimap$ and $\forall$ are restricted so that they do not occur to the left of the implication $\multimap$.

The reason that Lolli does not include LO is the presence of $\parr$ and $\bot$ in the latter. This suggests the following definition for Forum, the intended superlanguage: allow formulas to be freely generated from $\top$, $\&$, $\bot$, $\parr$, $\multimap$, $\Rightarrow$, and $\forall$. For various reasons, it is also desirable to add the modal ? directly to this list of connectives. Clearly, Forum contains the formulas in all the above logic programming languages.

Since the logics underlying Prolog, λProlog, Lolli, LO, and Forum differ in what logical connectives are allowed, richer languages modularly contain weaker languages. This is a direct result of the cut-elimination theorem for linear logic. Thus a Forum program that does not happen to use $\bot$, $\parr$, $\multimap$, and ? will, in fact, have the same cut-free proofs as are described for λProlog. Similarly, a program containing just a few occurrences of these connectives can be understood as a λProlog program that takes a few exceptional steps, but otherwise behaves as a λProlog program.

Forum is a presentation of all of linear logic since it contains a complete set of connectives. The connectives missing from Forum are directly definable using the following logical equivalences.

$$B^{\bot} \equiv B \multimap \bot \qquad 0 \equiv \top \multimap \bot \qquad 1 \equiv \bot \multimap \bot$$
$$!B \equiv (B \Rightarrow \bot) \multimap \bot \qquad B \oplus C \equiv (B^{\bot} \,\&\, C^{\bot})^{\bot} \qquad B \otimes C \equiv (B^{\bot} \parr C^{\bot})^{\bot}$$
$$\exists x.B \equiv (\forall x.B^{\bot})^{\bot}$$

The collection of connectives in Forum are not minimal. For example, ? and $\parr$, can be defined in terms of the remaining connectives.

$$?B \equiv (B \multimap \bot) \Rightarrow \bot \quad \text{and} \quad B \parr C \equiv (B \multimap \bot) \multimap C$$

The proof that Forum is, in fact, a logic programming language requires showing that if a sequent has a proof in linear logic, it has a proof that is, in a certain formal sense, *goal directed*. The proof of this follows from a result of Andreoli on focusing proofs [1]: for details, see [16].

Applications of Forum. Forum has been used to give specifications in various domains. We list some major extended examples below.

1. Forum can be used as a *logical framework* for the specification of both natural deduction and sequential calculus proof systems. Single conclusion, intuitionistic based systems, such as λProlog and LF [10], have been used to provide satisfactory treatments of natural deduction proof systems [7, 20, 22] but the specifications of sequent calculus in these systems is less than satisfactory [7, 21]. In [16], the multiple conclusion aspects of Forum allowed for natural and flexible presentations of sequent calculus, and Gentzen's LJ calculus [8] is studied in some depth there.

2. In [14], the operational semantics of the π-calculus [18] is specified. Important notions like scope extrusion, mobility, and testing are given simple proof theoretic treatments there.
3. The operational semantics of programming languages provides another area where Forum provides natural specifications.
   (a) Linear logic provides a simple mechanism for modeling state. Algol-like references and state is specified in [15, 16] and in this thesis, [5], Chirimar shows how the richer notion of state in Standard ML can be expressed.
   (b) Concurrency primitives similar to those found in Concurrent ML [23] are presented in [15, 16].
   (c) In [5] Chirimar presents specifications of exceptions and continuations (similar to those found in some implementations of SML). He also shows that these can be added modularly to his specification of call-by-value evaluation and state. With these specifications, he proves various equivalences among program phrases involving references and higher-order features.
   (d) Cervesato and Pfenning [4] similarly specify and analyzed ML-style references in a linear logic language that is similar to the Lolli subset of Forum.
4. Also in his thesis, Chirimar specifies both the sequential and the concurrent (piped-line) operational semantics of the DLX RISC processor [11]. He is able to capture the call-forwarding and early branch resolution optimizations and proves them to be equivalent. He also studies the problem of code equivalence via the Forum specification, and, in particular, analyzes the problem of code rescheduling for DLX.
5. In [3, 6], Bugliesi, Delzanno, Liquori, and Martelli specify and analyze an object-oriented programming language using Forum.

It should also be added that the area of natural language parsing should provide lots of other examples of where linear logic can be used to advantage. For example, Josh Hodas used Lolli to provide a declarative treatment of Filler-Gap Dependency Parsers [12]. See also a similar project in [19].

References

1. Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.
2. J.M. Andreoli and R. Pareschi. Linear objects: Logical processes with built-in inheritance. *New Generation Computing*, 9(3-4):445–473, 1991.
3. Michele Bugliesi, Giorgio Delzanno, Luigi Liquori, and Maurizio Martelli. A linear logic calculus of objects. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*. MIT Press, September 1996.

4. Iliano Cervesato and Frank Pfenning. A linear logic framework. In *Proceedings, Eleventh Annual IEEE Symposium on Logic in Computer Science*, pages 264–275, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.
5. Jawahar Chirimar. *Proof Theoretic Approach to Specification Languages*. PhD thesis, University of Pennsylvania, February 1995. Available on the web from `http//www.cis.upenn.edu/~dale/forum/`.
6. Giorgio Delzanno and Maurizio Martelli. Objects in forum. In *Proceedings of the International Logic Programming Symposium*, 1995.
7. Amy Felty. Implementing tactics and tacticals in a higher-order logic programming language. *Journal of Automated Reasoning*, 11(1):43–81, August 1993.
8. Gerhard Gentzen. Investigations into logical deductions. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland Publishing Co., Amsterdam, 1969.
9. Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
10. Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.
11. J. Hennesy and D. Patterson. *Computer Architecture A Quantitative Approach*. Morgan Kaufman Publishers, Inc., 1990.
12. Joshua 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, 1992.
13. Joshua Hodas and Dale Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.
14. Dale Miller. The π-calculus as a theory in linear logic: Preliminary results. In E. Lamma and P. Mello, editors, *Proceedings of the 1992 Workshop on Extensions to Logic Programming*, number 660 in LNCS, pages 242–265. Springer-Verlag, 1993.
15. Dale Miller. A multiple-conclusion meta-logic. In S. Abramsky, editor, *Ninth Annual Symposium on Logic in Computer Science*, pages 272–281, Paris, July 1994.
16. Dale Miller. Forum: A multiple-conclusion specification language. *Theoretical Computer Science*, 165:201–232, 1996.
17. Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
18. Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Part I. *Information and Computation*, pages 1–40, September 1992.
19. Fernando C. N. Pereira. Prolog and natural-language analysis: into the third decade. In *Proceedings of the 1990 North American Conference on Logic Programming*. MIT Press, 1990.
20. Frank Pfenning. Logic programming in the LF logical framework. In Gérard Huet and Gordon D. Plotkin, editors, *Logical Frameworks*. Cambridge University Press, 1991.

21. Frank Pfenning. Structural cut elimination. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 156–166, San Diego, California, 26–29 1995. IEEE Computer Society Press.
22. Frank Pfenning and Ekkehard Rohwedder. Implementing the meta-theory of deductive systems. In *Proceedings of the 1992 Conference on Automated Deduction*, June 1992.
23. John H. Reppy. CML: A higher-order concurrent language. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 293–305, June 1991.