LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - The University of Edinburgh

A Synopsis on the Identification of Linear Logic Programming Languages

by

James Harland & David Pym

LFCS Report Series **ECS-LFCS-92-248**
LFCS **November, 1992**
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Copyright © 1992, LFCS

Copyright © 1992, Laboratory for Foundations of Computer Science
University of Edinburgh. All rights reserved.

Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.

A Synopsis on the Identification of Linear Logic Programming Languages (Extended Abstract)[1]

James Harland
University of Melbourne
Australia

David Pym
University of Edinburgh
Scotland, U.K.

We investigate the definition of logic programming languages within linear logic [Gir87]. We take as our point of conceptual departure the *uniform proofs* of Miller *et al.*, [MNPS91], [Mil89] and the class of hereditary Harrop formulae of intuitionistic logic, although our aims are more foundational. In particular, we assume that the characteristic feature of logic programming is goal-directed proof-search. More precisely, there is a search operation corresponding to each logical connective, and when searching for a proof of a given goal one applies the search operation that corresponds to the outermost connective of that goal, and then to the outermost connective of each subgoal so generated, *etc.*. Furthermore, we assume that it must be possible to rewrite the program so that just one left rule, an appropriate notion of *resolution*[2] rule is required.

In *linear logic*, the identification of a class of computationally appealing proofs, uniform proofs, is somewhat more intricate than in logics that have been considered previously. For example, in first-order *intuitionistic* or *minimal* hereditary Harrop formulae [MNPS91], [Mil89], the search operation corresponding to each connective that can occur in a goal is given by the right rule for that connective. It turns out that for the whole intuitionistic hereditary Harrop fragment, in which just $\wedge$, $\supset$ and $\forall$ may occur negatively (the *definite formulae*) and $\wedge$, $\vee$, $\supset$, $\forall$ and $\exists$ may occur positively (the *goal formulae*), the strategy of constructing proofs (from root to leaves) by applying a right rule wherever one is applicable is complete. This property arises from the permutability properties of the rules of the (classical and) intuitionistic sequent calculus first studied by (Curry and) Kleene [Kle52, MNPS91, HP91, HP92].

The main novelty in linear logic is that for the desired classes of definite formulae and goal formulae, it may be necessary, under certain circumstances, to apply a left rule in the middle of a sequence of right rules in order to maintain completeness. In particular, we need to allow the $\otimes$-L and C!-L (contraction on the left) rules to occur immediately after the $\otimes$-R rule, and various left rules after the !-R rule [HP91, HP92]. One solution to this problem would be to exclude $\otimes$ and ! from the class of definite formulae in linear logic. However, this is both undesirable and unnecessary. It turns out that these exceptions to the strategy of applying right rules wherever they are applicable can be eliminated in the next step of our analysis, in which, just as in the setting of intuitionistic hereditary Harrop formulae, we rewrite the antecedents to a certain *clausal form* for which a single left rule, the appropriate notion of resolution rule, is complete.

[1]This work was supported in part by ESPRIT BRA, "Logical Frameworks"; and U.K. SERC grant GR/G 58588, "Logical and Semantical Frameworks"; and by a grant of the Australian Research Council.

[2]We stress that our notion of resolution is an *analytic* one; *cf.* classical Horn clause resolution, in which resolution amounts to cut together with unification.

Thus we show in [HP91, HP92] how an analysis of the permutability properties of the two-sided linear sequent calculus can be used to determine fragments of linear logic for which a suitable notion of resolution proof can be defined. A summary of this analysis consists in essentially two, mutually dependent, steps: (i) The identification of the class of formulae such that the strategy of constructing proofs (considering rules to be reduction operators, in the sense of Kleene) [Kle68] by applying right rules wherever they are applicable is complete, subject to certain exceptions to this strategy, which must be handled, *i.e.*, made goal-directed, by the second step. This class of proofs, constructed by the application of the right rules wherever possible, subject to the exceptions to be handled by step (ii), is the appropriate class of *uniform* proofs; (ii) The exploitation of the structural properties of the antecedents of linear sequents so that exceptions to the right rule-first strategy can be eliminated and so that we can invoke a single left rule, a *resolution* rule whilst retaining (soundness and) completeness with respect to linear sequent calculus. This is achieved by introducing a mapping which reduces antecedents to multisets of *clauses*.

This analysis leads us to identify the following classes of formulae as being available for use as a linear logic programming language:

$$\begin{array}{lllll} \textit{Definite formulae} & D & ::= & A \mid \mathbf{1} \mid \bot \mid D \,\&\, D \mid D \otimes D \mid D \parr D \mid G \multimap A \mid \bigwedge x\,.\,D \mid\, !D \\ \textit{Goal formulae} & G & ::= & A \mid \mathbf{1} \mid \bot \mid \top \mid D^{\bot} \mid G \otimes G \mid G \oplus G \mid G \parr G \mid G \,\&\, G \\ & & & \quad \mid D \multimap G \mid \bigwedge x\,.\,G \mid \bigvee x\,.\,G \mid\, !G \mid\, ?G, \end{array}$$

where A ranges over atomic formulae. Programs are linear antecedents that consist of just closed definite formulae and goals are linear succedents that consist of just closed goal formulae.

Of course, there are other, more pragmatically motivated, influences on the choices of definite formulae, and indeed on the choices of goal formulae. We remark here that if we were to restrict definite formulae to be just the clauses permitted by (ii), above, we should be forced to constrain goal formulae rather more than is essential.

It remains an open problem to determine precisely the maximal class of linear formulae for which suitable notions of goal-directed proof are complete. However, some upper and lower bounds may be given. The main omissions from the class of definite formulae are those of the form $D_1 \oplus D_2$, $\bigvee x.D$ and $?D$, as well as the constant $\mathbf{0}$. The first two cases should not be surprising; note that in the intuitionistic case, the formulae $D_1 \vee D_2$ and $\exists x.D$ are not allowed in definite formulae, as the relevant permutation properties do not hold otherwise. The same property holds in the linear case for $D_1 \oplus D_2$ and $\bigvee x.D$, and also for $?D$. The omission of $\mathbf{0}$ is also not peculiar to linear logic, but essentially a denial of the principle of *ex nihil quodlibet*; it seems difficult to reconcile the notion of goal-directed provability with the provability of $\Gamma, 0 \vdash \Delta$ for an arbitrary Δ. Thus it would seem that the omission of these four cases is necessary to preserve the notion of goal-directed provability.

The situation is less clear cut for some of the other omissions. Whilst the permutation properties appear to disqualify $G^{\bot}$ as a definite formula on the same grounds as $D_1 \oplus D_2$, $\bigvee x.D$ and $?D$, $G^{\bot}$ is linearly equivalent to $G \multimap \bot$, and as $\bot$ may appear in goals, it seems reasonable to suggest that formulae of the form $G \multimap \bot$ may be included in the class of definite formulae. This form of negated formulae may considered as a way of ensuring that

definite formulae containing negations may only be "used" at particular places in a proof.

Another "grey area" is for formulae of the form $G \multimap D$, rather than merely $G \multimap A$ where A is an atom. In the intuitionistic case, it is known that a formula of the form $G \supset D$ may be converted into an equivalent set of definite formulae in which only formulae of the form $G \supset A$ appear. However, this property does not hold in linear logic — for example, $p \multimap (q \otimes r)$ is not linearly equivalent to $(p \multimap q) \otimes (p \multimap r)$. Hence in the linear case, the addition of formulae of the form $G \multimap D$ increases both expressibility and power. The main reason that formulae of the form $G \multimap A$ are desirable is that they allow *simple* proofs to be used, *i.e.,* that the $\supset$-L rule need only be used as a unary rule, which in turn leads to the familiar notion of unifying an atom with the head of a clause. Clearly such a notion of proof cannot be used in the presence of formulae of the form $G \multimap D$. However, uniform provability remains complete for such formulae, and so we may choose to maintain either simple proofs or formulae of the form $G \multimap D$, but not both. In this sense the notion of maximality will depend upon whether we consider goal-directed provability to include the notion of simple proof or not. Another possibility is to restrict D to a subclass of definite formulae with a correspondingly restricted class of subproofs appearing on the right of $\multimap$-L, such as allowing the subproof to contain only the rules $\parr$-L, & -L and $\otimes$-L. Such a straightforward modification of our notion of uniform proof permits the presence of definite formulae of the form $G \multimap (A_1 \parr \ldots \parr A_m)$, for example. This seems to be an interesting compromise between simple proofs and allowing arbitrary (uniform) proofs on the right of $\multimap$-L; in particular, no right rules or occurrences of $\multimap$-L will appear in such subproofs, and so the subproof resembles a particularly simple form of clausal decomposition. Whatever choice is made, it is clear that the answer to the maximality question depends very much on the the way in which we allow the $\multimap$-L rule to be used.

Hence the above class of formulae may not be the "ultimate" linear logic programming language, but it would seem that it is not very far from it, and that it will be a subtle and/or intricate task to expand the language in significant ways. The more abstract problem of identifying, for a given class of proofs (such as uniform proofs), the maximal class of formulae for which that class of proofs is complete with respect to provability remains an intriguing open problem and the subject of active research.

Whilst resolution proofs determine the nature of the search primitives needed to mechanically implement this language as far as is possible *logically*, they are not, in *operational* terms, deterministic. In order describe *interpreters* we must have a suitably deterministic, operational characterization of proofs. For example, consider the $\otimes$-rule of [HP92]:

$$\frac{\mathcal{D}_0 \vdash_R G_1, \mathcal{G}_0 \quad \mathcal{D}_0' \vdash_R G_2, \mathcal{G}_0' \quad \mathcal{D}_1 \vdash_R \mathcal{G}_1 \quad \ldots \quad \mathcal{D}_n \vdash_R \mathcal{G}_n}{\mathcal{D} \vdash_R G_1 \otimes G_2, \mathcal{G}},$$

where $C \in \mathcal{D}$ and $\{\mathcal{D}_0, \mathcal{D}_0' \vdash_R G_1 \otimes G_2, \mathcal{G}_0, \mathcal{G}_0'\} \cup \bigcup_{i=1}^{n} \{\mathcal{D}_i \vdash_R \mathcal{G}_i\}$ is a multiset of *C-components* of the sequent $\mathcal{D} \vdash_R G_1 \otimes G_2, \mathcal{G}$. In this rule, read from conclusion to premisses, the program $\mathcal{D}$ and goal $\mathcal{G}$ are, roughly speaking, split, via C-components, into the parts $\mathcal{D}_0, \mathcal{D}_0', \mathcal{D}_1, \ldots, \mathcal{D}_n$ and $\mathcal{G}_0, \mathcal{G}_0', \mathcal{G}_1, \ldots, \mathcal{G}_n$, respectively. The reader might easily verify that the following is an example of the $\otimes$-rule:

$$\frac{!(p \,\&\, q), p \vdash_R p \quad !(p \,\&\, q), p \vdash_R p \quad !(p \,\&\, q), q \vdash_R q \quad !(p \,\&\, q), q \vdash_R q}{!(p \,\&\, q) \vdash_R p, p, q \otimes q}. \tag{1}$$

But the rule provides no determination of how this splitting is to be calculated. A similar problem arises with resolution rule in linear logic programming [HP92]: a clause may need to be deleted after it has been used, *i.e.*, after a resolution step has been performed. Indeed, on consideration of the resolution rule (below), the reader will see that both of these issues arise:

$$\frac{\mathcal{D}_1 \vdash_R \mathcal{G}_1 \ \ldots \ \mathcal{D}_n \vdash_R \mathcal{G}_n}{\mathcal{D} \vdash_R A, \mathcal{G}},$$

where $\bigcup_{i=1}^{n} \{\mathcal{D}_i \vdash_R \mathcal{G}_i\}$ is a *resolvant* of $\mathcal{D} \vdash_R A, \mathcal{G}$: a resolvant[3], for a particular choice of clause $G \multimap A \in \mathcal{D}$ which does not appear in $\mathcal{D}_1, \ldots, \mathcal{D}_n$, determines how to split $\mathcal{D}$ ($\mathcal{G}$) into $\mathcal{D}_1, \ldots, \mathcal{D}_n$ ($\mathcal{G}_1, \ldots, \mathcal{G}_n$).

The definition of the resolution rule is not as complicated as the reader might at first suppose. For example, the reader might easily check that the following, in which we write c for the formula $!((p \multimap r) \,\&\, ((q_1 \otimes q_2) \multimap s))$, is an instance of the resolution rule (*2*) (above):

$$\frac{c, p, p \multimap r \vdash_R r \qquad c, q_1, q_2, (q_1 \otimes q_2) \multimap s \vdash_R q_1 \otimes q_2}{p, q_1, q_2, !((p \multimap r) \,\&\, ((q_1 \otimes q_2) \multimap s)) \vdash_R r, s}, \tag{2}$$

Again, the rule provides no determination of how this splitting is to be calculated.

Our solution [HP91, HP92], at the abstract level, to the problem of how to calculate splittings is to adopt a *lazy* approach, and to this end we permit an interpreter to construct *proto-proofs* by modifying the rules of resolution proof to delay the calculation of splittings (so that, for example, all suitable formulae in the antecedent and succedent go each way at a $\otimes$-rule). In order to maintain the soundness, we introduce the notion of *path* in such proto-proofs. Paths can be considered to be *proto-proof-objects* — they are the appropriate notion of proof-object for our computational purposes and are related to *proof-nets* [Gir87].

We sketch the definition of the construction of a *path*[4] in such a proto-proof as follows: (I) The endsequent is in every path; (II) Traverse the proto-proof tree towards the leaves, starting at the endsequent: (i) Whenever a &-rule is reached, choose a branch and proceed; (ii) Whenever a $\otimes$- or resolution rule is reached, proceed along all branches; (III) Continue until all branches of the path have reached a leaf. The proto-proof determines a proof just in case for each possible path in it, there are expansions (at the appropriate rule applications) of the antecedent and succedent that are compatible with leaves in the path.

We illustrate the notion of path with an example of a proto-proof involving both the $\otimes$- and resolution rules. (As above, we denote the formula $!((p \multimap r) \,\&\, ((q_1 \otimes q_2) \multimap s))$ as c where convenient.)

$$\dfrac{\dfrac{c, p, q_1, q_2 \vdash^{\mathbf{I}}_R p}{c, p, q_1, q_2, p \multimap r \vdash^{\mathbf{I}}_R r}\ \text{res.} \qquad \dfrac{c, p, q_1, q_2, (q_1 \otimes q_2) \multimap s \vdash^{\mathbf{I}}_R q_1 \quad c, p, q_1, q_2 \vdash^{\mathbf{I}}_R q_2}{c, p, q_1, q_2, (q_1 \otimes q_2) \multimap s \vdash^{\mathbf{I}}_R q_1 \otimes q_2}\ \otimes}{p, q_1, q_2, !((p \multimap r) \,\&\, ((q_1 \otimes q_2) \multimap s)) \vdash^{\mathbf{I}}_R r, s}\ \text{res.}$$

[3] In a simple propositional setting.

[4] Such a construction proceeds dynamically, during the construction of a proto-proof.

There is just one path (marked by ♯) in this proto-proof and the reader might easily verify that suitable splittings and expansions[5] exist for a resolution proof to be determined, *i.e.*, that there is indeed a resolution proof of this endsequent. To see this, consider that the leaves on the path require exactly one occurrence of each of q_1, q_2 and p; that the implicational subformulae of c are each used exactly once; and that these conditions are compatible with the proto-proof. It should be noted that the input/output model of [HM91] can be considered to be a particular computational method of determining paths.

The full theory of paths, which has much in common with the work of Andrews, Bibel and Wallen on the use of matrices and matings for proof-search [And81, Bib82, Wal89], remains to be developed in detail. Such a theory provides the transition from the logical characterization of goal-directed proof-search given by resolution proofs to a truly operational account, whilst making no inessential commitment to a particular implementation strategy.

We remark that it will be important and interesting to study broader classes of relevance logics.

References

[And81] Andrews, P.B. *Theorem-proving via general matings.* J. Assoc. Comp. Mach. 28(2):193-214, 1981.

[Bib82] Bibel, W. *Computationally Improved Versions of Herbrand's Theorem.* Logic Colloquium '81, pp. 11-28. North-Holland, 1982.

[Gir87] Girard, J.-Y. *Linear logic.* Theoretical Computer Science, 50:1–102, 1987.

[HP91] Harland, J., Pym, D. *The Uniform Proof-theoretic Foundation of Linear Logic Programming (Extended Abstract).* Proceedings of the International Logic Programming Symposium (V. Saraswat and K. Ueda, editors), San Diego, October, 1991. MIT Press, 1991. pp. 304-318.

[HP92] Harland, J.A., Pym, D.J. *Resolution in Fragments of Classical Linear Logic (Extended Abstract).* Proceedings of the Russian Conference on Logic Programming and Automated Reasoning (A. Voronkov, editor), St. Petersburg, July, 1992. Lecture Notes in Artificial Intelligence, Vol. 624, Springer-Verlag, 1992. pp. 30-41.

[HM91] Hodas, J., Miller, D. *Logic Programming in a Fragment of Intuitionistic Linear Logic: Extended Abstract.* Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science, Amsterdam, July, 1991. IEEE Computer Society Press, 1991. pp. 32-42.

[Kle68] Kleene, S.C. *Mathematical Logic.* Wiley and Sons, 1968.

[Kle52] Kleene, S.C. *Permutability of inferences in Gentzen's calculi LK and LJ.* Memoirs of the American Mathematical Society **10**, pp. 1-26, 1952.

[Mil89] Miller, D. *A Logical Analysis of Modules in Logic Programming.* J. Logic Prog. 6(1& 2), 1989, pp. 79-108.

[MNPS91] Miller, D., Nadathur, G., Pfenning, F., Ščedrov, A. *Uniform Proofs as a Foundation for Logic Programming.* Annals of Pure and Applied Logic 51, pp. 125-157, 1991.

[Wal89] Wallen, L.A. *Automated Deduction in Non-classical Logics.* MIT Press, 1989.

[5] Multiplicities of formulae, such as c, with ! as their outermost connective.