The Uniform Proof-theoretic Foundation....

The Uniform Proof-theoretic Foundation of Linear Logic Programming (Extended Abstract)

by

James Harland
David Pym

LFCS Report Series ECS-LFCS-91-168

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

June 1991

The Uniform Proof–theoretic Foundation of Linear Logic Programming (Extended Abstract)

James Harland
Department of Computer Science
University of Melbourne
Parkville, 3052
Australia
jah@cs.mu.oz.au

David Pym
Department of Computer Science
University of Edinburgh
Edinburgh EH9 3JZ
Scotland, U.K.
dpym@lfcs.ed.ac.uk

Abstract

We present a proof-theoretic analysis of a natural notion of *logic programming* for Girard's *linear logic*. This analysis enables us to identify a suitable notion of *uniform proof*. This in turn enables us to identify choices of classes of definite and goal formulae for which uniform proofs are complete and so to obtain the appropriate formulation of *resolution proof* for such choices. Resolution proofs in linear logic are somewhat difficult to define. This difficulty arises from the need to decompose definite formulae into a form suitable for the use of the linear resolution rule, a rule which requires the selected clause to be *deleted* after use, and from the presence of the *modality* ! (of course).

We consider a *translation* — resembling those of Girard — of the intuitionistic hereditary Harrop formulae and intuitionistic uniform proofs into our framework, and show that certain properties are preserved under this translation.

We sketch the design of an *interpreter* for linear logic programs.

1 Introduction

An interesting recent development in logic of some significance for theoretical computer science is *linear logic* [8].

A natural question which arises is to determine what significance this development may have for logic programming; in particular, the identification of a particular set of formulae as a linear logic programming language, and the characterization of the novel features of a linear logic programming language.

In this paper we show how linear logic programming languages may be identified, purely by a proof-theoretic analysis. This is done by studying certain permutation properties of a sequent calculus presentation of linear logic. This in turn allows us to identify classes of formulae for which a smaller class of proofs, known as *uniform proofs*, can be shown to be complete. Uniform proofs have been studied in the past [13], [12], and it seems that such proofs characterize the proof theory of logic programming. As it is easy to show that uniform proofs are not complete for full linear logic, we thus identify linear logic programming languages by classes of formulae for which uniform proofs are complete.

Furthermore, we show how a linear logic version of the resolution rule may be given. The major difference in this case is that the clause used in the resolution step is deleted from the program. This allows the programmer to specify an upper bound for the number of times a given clause is to be used, and we may think of the resolution step in linear logic to mean "resolve and retract". From a programming point of view, this property allows the specification, in a logical manner, of features such as destructive assignment and computation bounded by time and/or space.

However we need not be restricted to this rule, as linear logic contains modal operators which allow intuitionistic logic to be embedded within it, and so the standard resolution rule may be used under certain circumstances. Hence our linear logic programming language allows these two versions of the resolution rule to be used in the one system.

We take as our point of departure the notion of *uniform proof*, introduced by Miller *et al.* [13], [12]. We present a proof-theoretic analysis of this notion and the attendant notions of definite and goal formulae for linear logic. In Section 2 we develop the appropriate notion of uniform proof for linear logic and proceed to show how classes of definite and goal formulae may be defined. Section 3 shows how resolution works in this context, and in Section 4 we discuss some issues pertaining to the implementation of an interpreter for our language. Finally in Section 5 we show how the system of Miller *et al.* may be embedded in ours, thus showing our system to be a conservative extension of the standard ones.

A preliminary account of this work appears in [9].

2 Uniform Linear Proofs

We assume a standard two-sided presentation of linear sequent calculus [8], [3], without the constants of linear logic, as presented in Appendix A.

One of the distinguishing features of logic programming is goal-directed 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.*. For Horn clause goals, *i.e.*, existentially quantified conjunctions of atoms, this corresponds to using unification to "delay" the choice of witnesses for the variables, and a strategy for selecting a particular atom as the next subgoal. For a larger language, such as first-order hereditary Harrop formulae [13], we need a richer set of primitive search operations, or *search primitives*, so that each distinct connective corresponds to a distinct search primitive.

Clearly it is not difficult to identify the appropriate search operation for a given right rule, and indeed it is clear that such a procedure will be sound. However, in addition, we need to know that this method of searching for proofs is complete: that we are guaranteed not to miss a consequence. Hence our problem is not so much to identify the required search primitives for the right rules of linear logic as to show that the goal-directed search strategy is complete.

Such considerations imply that a methodology for determining classes of formulae for which this strategy is complete is in a turn a methodology for finding logic programming languages. Hence, in order to determine what classes of formulae may be considered as linear logic programming languages, we need to consider a particular class of proofs which reifies such a strategy.

In [13] it is shown how *uniform proofs* are complete for a certain fragment of (intuitionistic) first-order logic. A uniform proof in that context is one in which the outermost connective of the succedent is introduced in the previous step. Thus, in terms of goal-directed search, this means that right rules are applied as early as possible in the proof search process. The definition of such uniform proofs in linear logic is somewhat more intricate, the main difficulty being that, under certain circumstances, it may be necessary to apply a left rule in the middle of a sequence of right rules in order to maintain completeness. For example, consider the sequent $\phi \otimes \psi \vdash \phi \otimes \psi$. Clearly there is a proof in which the $\otimes$-R rule precedes the $\otimes$-L rule, *i.e.*,

$$\dfrac{\dfrac{\phi \vdash \phi \qquad\qquad \psi \vdash \psi}{\phi, \psi \vdash \phi \otimes \psi}\ \otimes\text{-R}}{\phi \otimes \psi \vdash \phi \otimes \psi}\ \otimes\text{-L}$$

but we cannot apply the rules in the reverse order, as neither $\vdash \phi$ nor $\phi \otimes \psi \vdash \psi$ is provable. Hence, we cannot always push the $\otimes$-L above the $\otimes$-R rule; but the cases in which this is not possible may be classified.

A similar example occurs for the $C!$-L rule. Consider the sequent $!\phi \vdash \phi \otimes \phi$. Clearly there is a proof in which the $\otimes$-R rule precedes the $C!$-L rule, *i.e.*,

$$\dfrac{\dfrac{\dfrac{\phi \vdash \phi}{!\phi \vdash \phi}\ !\text{-L} \qquad \dfrac{\phi \vdash \phi}{!\phi \vdash \phi}\ !\text{-L}}{!\phi, !\phi \vdash \phi \otimes \phi}\ \otimes\text{-R}}{!\phi \vdash \phi \otimes \phi}\ C!\text{-L}$$

but we cannot apply the rules in the reverse order, as $\vdash \phi$ is not provable. Thus, as in the previous case, we cannot always push the $C!$-L above the $\otimes$-R rule; but the cases in which this is not possible may be classified.

As it happens, these are the only two such cases; in all the others the "outermost connective first" strategy will suffice. We develop formally the notion of a uniform proof in this context.

In what follows $*$-R and $\sharp$-L range respectively over the right and left rules for the linear connectives as well as the rules $W?$-R, $W!$-L, $C?$-R and $C!$-L.

Definition 2.1 (RL, LR and Locally LR Proofs) *We define the following shapes of linear proofs:*

1. *A proof is* RL *if there is an occurrence of a right rule preceding a left rule;*
2. *A proof is* LR *if all occurrences of right rules appear after all left rules;*
3. *A proof is* locally LR *if the only occurrences of a right rule preceding a left rule are either of the form:*

$$\dfrac{\dfrac{\Xi_1}{\Gamma \vdash \phi, \Delta}\ *\text{-R} \qquad \dfrac{\Xi_2}{\Gamma', \psi \vdash \Delta'}\ \sharp\text{-L}}{\Gamma, \Gamma', \phi \multimap \psi \vdash \Delta, \Delta'}\ \multimap\text{-L}$$

where Ξ_1 and Ξ_2 are locally LR, *or of the form:*

$$\dfrac{\dfrac{\dfrac{\Xi_1}{\Gamma \vdash \phi, \Delta} \qquad \dfrac{\Xi_2}{\Gamma' \vdash \psi, \Delta'}}{\Gamma, \Gamma' \vdash \phi \otimes \psi, \Delta, \Delta'}\ \otimes\text{-R}}{\Gamma'', \chi \otimes \xi \vdash \phi \otimes \psi, \Delta, \Delta'}\ \otimes\text{-L}$$

where $\chi \in \Gamma$, $\xi \in \Gamma'$, and Ξ_1 and Ξ_2 are locally LR, *or of the form:*

$$\dfrac{\dfrac{\dfrac{\Xi_1}{\Gamma \vdash \phi, \Delta} \qquad \dfrac{\Xi_2}{\Gamma' \vdash \psi, \Delta'}}{\Gamma, \Gamma' \vdash \phi \otimes \psi, \Delta, \Delta'} \otimes\text{-R}}{\Gamma'', !\chi \vdash \phi \otimes \psi, \Delta, \Delta'} \; C!\text{-L}$$

where $!\chi \in \Gamma$, $!\chi \in \Gamma'$, *and* Ξ_1 *and* Ξ_2 *are locally* LR. □

However, locally LR proofs are not quite satisfactory for our purposes, as they might involve some inessentially complicated subproofs on the right hand side of the $-\!\circ$-L rule. Hence we introduce the notion of a *simple* proof, *cf.* [12].

The intuition behind simple proofs is that the right hand subproof used in the $-\!\circ$-L rule should be trivial, *i.e.*, require no effort to prove. For example the following proof is not simple:

$$\dfrac{\phi \vdash \phi \qquad \dfrac{\phi \vdash \phi \qquad \psi \vdash \psi}{\phi, \phi -\!\circ\, \psi \vdash \psi} -\!\circ\text{-L}}{\phi, \phi -\!\circ\, \phi, \phi -\!\circ\, \psi \vdash \psi} -\!\circ\text{-L}$$

However, it is clear that the sequent $\phi, \phi -\!\circ\, \phi, \phi -\!\circ\, \psi \vdash \psi$ does have a simple proof. This property in fact holds for any sequent in which the antecedent is a program and the consequent a goal.

Definition 2.2 (Simple Proofs) *A linear proof* Ξ *is said to be simple if every occurrence in* Ξ *of the* $-\!\circ$-L *rule is of the form:*

$$\dfrac{\begin{matrix}\Xi' \\ \Gamma \vdash \phi, \Delta\end{matrix} \qquad \psi \vdash \psi}{\Gamma, \phi -\!\circ\, \psi, \Gamma' \vdash \psi, \Delta, \Delta'}$$

□

Henceforth, we shall call simple locally LR proofs *uniform* proofs.

Our aim here is to show how, with the linear rules read as *reduction operators* from conclusion to premisses[1], it is possible to obtain goal-directed proof-search procedures for certain classes of linear formulae. To this end, we identify classes of definite and goal formulae for which uniform proofs are complete for consequence. There are several choices which may be made at this point, and in Definition 2.3 below we present illustrative classes of formulae rather than large ones. For example, we limit negation to be applied only to atoms; extensions to the class of negated formulae are possible. Also, for simplicity, we limit implicational definite formulae to exclude $G -\!\circ (A_1 \,\&\, \ldots \,\&\, A_m)$, $G -\!\circ \bigwedge \vec{x}\,.\,A$, *etc.*, see [10]. Some other extensions are mentioned at the end of this section.

[1]Kleene [11] explains this in the case of classical predicate calculus.

Definition 2.3 (Definite and Goal Formulae) *Let A range over atomic formulae. We define classes of* definite formulae *and* goal formulae *as follows:*

$$\begin{array}{llll} \textit{Definite formulae} & D & ::= & A \mid D \,\&\, D \mid D \otimes D \mid G \multimap A \mid \bigwedge x \,.\, D \mid !D \\ \\ \textit{Goal formulae} & G & ::= & A \mid A^{\perp} \mid G \otimes G \mid G \oplus G \mid G ⅋ G \mid G \,\&\, G \\ & & & \mid D \multimap G \mid \bigwedge x \,.\, G \mid \bigvee x \,.\, G \end{array}$$

Programs *are linear antecedents that consist of just closed definite formulae and* goals *are linear succedents that consist of just closed goal formulae. We assume that all quantified variables are* standardized apart. □

The resemblance of these definitions to those of hereditary Harrop formulae of intuitionistic logic should be clear. The reader who is unfamiliar with linear logic should note that full linear logic has four constants, whose ontological status is akin to that of $\perp$ in intuitionistic logic, and with which are associated special axioms and rules. We do not analyse the constants and their rules here: such an analysis is possible.

An important point to note is that, unlike intuitionistic logic, in a given sequent $\Gamma \vdash \Delta$, the antecedent Γ and succedent Δ need to be considered as *multisets* of formulae rather than sets. This difference is due to the lack of the (general) weakening and contraction rules in linear logic, which are present in intuitionistic (and classical) logic. These properties may be recovered in certain cases by the use of the operators ! and ? and the $W!$, $W?$, $C!$ and $C?$ rules, which hence provide the ability to embed intuitionistic logic in linear logic. Furthermore, it is important to note that linear antecedents are proof-theoretically characterized by the tensor product ($\otimes$) of their components and that linear succedents are characterized by the tensor sum ($⅋$) of their components. We note that by considering the antecedents and succedents of linear sequents to be multisets of formulae we are able to suppress all occurrences of the *exchange* rules in linear proofs. If $\mathcal{F} = \{\phi_1, \ldots, \phi_m\}$ is a multiset of linear formulae then we write $\bigotimes_{\phi \in \mathcal{F}} \phi$ to denote the formula $\phi_1 \otimes \ldots \otimes \phi_m$.

It is somewhat tedious but not very difficult to show the following result:

Theorem 2.4 *Let Γ be a multiset of definite formulae, and let Δ be a multiset of goal formulae. Then $\Gamma \vdash \Delta$ has a linear proof if and only if $\Gamma \vdash \Delta$ has a uniform (or simple locally LR) proof.* □[2]

As mentioned earlier, there are larger classes of formulae for which our analysis will hold. In particular, the analysis above still applies when $?G$ is allowed as a goal. In addition, goals of the form $!G$ may be used, in which case a slightly more intricate notion of locally LR proofs is needed. In [10], restrictions are placed on definite formulae to permit goal formulae of this form. Both of these extensions require additional detail rather than fundamentally new features. A more significant extension, again for which the above analysis remains germane, is the use of definite formulae of the form $D_1 ⅋ D_2$. The reason that this is more involved is that it requires an extension to the way that programs are used during computation, *i.e.* to the resolution rule and the notion of a clause. For simplicity we do not pursue this or other extensions here; however it should be noted that our analysis thus far applies to these other cases. These and other issues will be dealt with in a forthcoming paper by the authors.

[2]This result amounts to a *Permutation Theorem* for the given fragment of linear logic. The reader who is interested in Permutation Theorems in general is referred to [5] for classical logic, [15] for a variety of non-standard first-order connectives, and [14] for a constructive dependent type theory.

3 Resolution

We saw in the previous section how simple locally LR proofs are complete for sequents $\mathcal{P} \vdash \mathcal{G}$ where $\mathcal{P}$ is a multiset of definite formulae (or program) and $\mathcal{G}$ is a multiset of goal formulae (or goal). In this section we show how a more specific class of proofs, which we call *resolution* proofs, may be used in a similar fashion. We present this analysis for the definite formulae and goal formulae of Definition 2.3: it can be extended to the additional classes of such formulae discussed above. The important characteristics of such resolution proofs are that they are goal-directed and that they use a single left rule, namely *resolution*[3].

First we introduce the notions of *clause* and *clausal decomposition* of a program $\mathcal{P}$.

Definition 3.1 (Clause) *A* linear clause *is a formula of the form A or $G \multimap A$, where A ranges over atomic formulae and G ranges over goal formulae. A* mixed clause *is a formula of the form A, $G \multimap A$ or $!(C_1 \otimes \ldots \otimes C_n)$, where each C_i, $1 \leq i \leq n$, is a linear clause.* □

Note that when G is a conjunction of atoms, a linear clause in the above sense is just a *Horn* clause. Below we show how to generate a multiset of clauses from a multiset of definite formulae via the mapping [-].

We will find it convenient to divide an antecedent into two parts — those formulae which are of the form $!F$, and those which are not.

Definition 3.2 (Antecedent Division) *Let $\cup$ denote multiset union, and let $\mathcal{D}$ be a multiset of formulae. We define two multisets $\mathcal{D}^I$ and $\mathcal{D}^L$ such that $\mathcal{D} = \mathcal{D}^I \cup \mathcal{D}^L$ as follows:*

$$\mathcal{D}^I = \bigcup_{F \in \mathcal{D}} \{\ F \mid \textit{the outermost connective of } F \textit{ is } !\ \}$$

$$\mathcal{D}^L = \bigcup_{F \in \mathcal{D}} \{\ F \mid \textit{the outermost connective of } F \textit{ is not } !\ \} \ \square$$

We may think of this distinction as specifying which formulae are known to be able to be re-used (those commencing with !) and those which are not.

Definition 3.3 (Clausal Decomposition) *Let $\cup$ denote multiset union. We define a mapping $[-]$ from multisets of definite formulae to multisets of mixed clauses as follows:*

$$\begin{array}{lll@{\qquad}lll}
[\mathcal{P}] & =_{\text{def}} & \bigcup_{D \in \mathcal{P}} [D] & [G \multimap A] & =_{\text{def}} & \{G \multimap A\} \\
[A] & =_{\text{def}} & \{A\} & [\bigwedge x \,.\, D] & =_{\text{def}} & [D] \\
[D_1 \,\&\, D_2] & =_{\text{def}} & [D_1] \text{ or } [D_2] & [!\,D] & =_{\text{def}} & \{!\bigotimes_{C \in [D]^L} C\} \cup [D]^I \\
[D_1 \otimes D_2] & =_{\text{def}} & [D_1] \cup [D_2] & & &
\end{array}$$

where variables x that occur in $\mathcal{P}$ and outwith the scope of some ! are marked as global *and where variables x that occur in $\mathcal{P}$ within the scope of some ! are marked as* local. □

[3]In this presentation, the resolution rule is split into two cases: in the purely linear fragment there would be just one. Such a split is inessential and a single rule will suffice provided one defines a suitable notion of *resolvant* for mixed clauses (Definition 3.1). Indeed, such an approach is desirable if one considers definite formulae of the form $D_1 \parr D_2$, *etc.*, which would otherwise require further cases of the resolution rule.

Note that $[\mathcal{P}]$ is a multiset of mixed clauses.[4] For example, let $\mathcal{P}$ be given by the definite formula $\bigwedge x \,.\, (p(x) \otimes (p(x) \multimap q(x)))$; then we have that $[\mathcal{P}]$ is the multiset $\{ p(x), p(x) \multimap q(x) \}$. Thus we may think of the transformation $[-]$ as a mapping which given a program, *i.e.* a multiset of definite formulae, yields a multiset of clauses which are the *compiled* form of the program, *i.e.* the form for which resolution proof can be defined. For given $\mathcal{P}$, $[\mathcal{P}]$ can be considered to be a *normal form* of $\mathcal{P}$.

Note that there are no quantifiers on the variables in the clauses in $[\mathcal{P}]$, although certain of the variables are marked as being "global" and certain of the variables are marked as being "local"; these are the variables that are quantified in $\mathcal{P}$. One point to note is that the mixture of universal quantifiers and ! may lead to some (possibly) surprising results. For example, let $\mathcal{P}$ be the single definite formula $\bigwedge x \,.\, ! \bigwedge y \,.\, p(x, y)$, and consider the goal $p(a, b) \otimes p(a, c)$. It should be clear that $! \bigwedge y \,.\, p(a, y) \vdash p(a, b) \otimes p(a, c)$ is provable, and so $\bigwedge x \,.\, ! \bigwedge y \,.\, p(x, y) \vdash p(a, b) \otimes p(a, c)$ is provable.[5] Hence we may think of universally quantified variables outside the scope of ! as "global", in that all occurrences of the variable must be updated consistently. Due to the possibility of splitting the program during computation, this may mean updating variables simultaneously across several branches of the proof.[6] On the other hand, the universally quantified variables within the scope of ! may be thought of as "local", in that they need only be updated in one formula. This leads us to the definitions that follow.

Definition 3.4 (Global and Local Variables, Distinctness) *Let $\mathcal{P}$ be a program. If x is a free variable in $[\mathcal{P}]$, then it is a* global variable *if it occurs outside the scope of any ! in P. Otherwise, x is a* local variable.

Let $\mathcal{D}$ be a multiset of definite formulae. We say $\mathcal{D}'$ and $\mathcal{D}''$ are distinct copies *of $\mathcal{D}$ if $\mathcal{D}'$ and $\mathcal{D}''$ are obtained from $\mathcal{D}$ by renaming the free variables of $\mathcal{D}$ so that $\mathcal{D}'$ and $\mathcal{D}''$ have no free variable names in common.* $\square$

Note that the marked "global" and "local" variables of Definition 3.3 satisfy this definition. For a given substitution $[\vec{t}/\vec{x}]$, we write $[\vec{t}/\vec{x}]^g$ to denote that the application of the substitution is to only those free variables that are global, throughout the proof. Otherwise, $[\vec{t}/\vec{x}]$ applies to all global variables, throughout the proof, and to local variables only in the sequent in which the substitution arises (and to those above it, of course).

One of the major differences between intuitionistic logic programming and linear logic programming is that in the latter case we may need to split the program (and indeed the goal), due to the form of $\otimes$-R rule. In the case of linear clauses, this is simply a matter of dividing up the given multiset. However for mixed clauses, we need a more sophisticated approach, as formulae beginning with a ! may be used any number of times in a proof. From the contraction rule we know that if $\mathcal{P}, !D, !D \vdash \mathcal{G}$ is provable, then so is $\mathcal{P}, !D \vdash \mathcal{G}$, and so when searching for an appropriate splitting of the program, we need to add a formula of the form $!D$ to each branch. We

[4]The decomposition [-] is related to the theory of *proof nets*, and indeed to the notions of *reduction ordering* — a combination of *subformula ordering* and *substitution ordering* — and *matrix methods*, described in [2], [4], [15] and [14]. Such constructions are useful in the study of Permutation Theorems, *q.v.* Theorem 2.4.

[5]*Cf.* the *Barcan formula*: $\forall x \Box A(x) \supset \Box \forall x\, A(x)$.

[6]In terms of the resolution proofs defined below, those branches that are above the same premiss of the last & –rule (Rule 8 in the definition of resolution proof, below) as the sequent to which the resolution rule is applied. The form of the &-rule in such proofs enforces the restriction of substitutions to their own &-branches.

will also need to "balance" the use of subformulae of $!D$, as we can only add an arbitrary number of copies of D to the antecedent, and not arbitrary numbers of a given subformula of D. For example, consider the program $!(p \otimes q)$ and the goal $p \otimes (q \otimes q)$.[7] By use of the contraction rule in conjunction with !-L and $\otimes$-L, the program will imply the goal if $p, q, !(p \otimes q) \vdash p \otimes (q \otimes q)$ is provable. If we reduce this to $p, !(p \otimes q) \vdash p$ and $q, !(p \otimes q) \vdash q \otimes q$, then the first sequent is provable, but the second is not, as we need to add p, q to the antecedent before splitting again, and we find that $p, q, !(p \otimes q) \vdash q$ is not provable. However, if we were allowed to duplicate more copies of the subformula q of $p \otimes q$ than of the subformula p we should be able to prove the goal $p \otimes (q \otimes q)$ from the program $!(p \otimes q)$. Hence we need to ensure that the splitting of the program is done in such as way as to respect the structure of the original formulae.

The reader should note however that as yet we have not provided any mechanism for the calculation of an appropriate splitting of the program (and goal). We postpone the description of such a procedure until we have defined the notion of resolution proof and are ready to describe an interpreter.

An essential feature of the mapping $[-]$ is that it builds into resolution proofs all necessary instances of the $\otimes$-L and $C!$-L rules. Therefore the exceptional cases in the definition of uniform (simple locally LR) proofs do not arise in resolution proofs. For example, as noted above, the sequent $p \otimes q \vdash p \otimes q$ is provable, and requires the use of the left rule "below" that of the right rule. However, $[p \otimes q] = \{p, q\}$, and so by replacing $p \otimes q$ with $\{p, q\}$ we eliminate the need for the left rule, and hence the need for an exception to be made to the "outermost connective first" strategy. A similar simplification may be made for the other exceptional case by allowing formulae commencing with a ! to be added to both branches of a proof containing an occurrence of the $\otimes$-R rule and appropriately modifying the notion of an initial sequent.

Definition 3.5 (Multiset Expansion) *Let C be a multiset and let $\cup$ denote multiset union. We define C^n for integers $n \geq 0$ inductively as follows:*
$C^0 = \emptyset \qquad C^{n+1} = C \cup C^n \qquad \square$

Definition 3.6 (Expansion) *Let $\mathcal{D}$ be a multiset of mixed clauses, and suppose we have that $\mathcal{D}^I = \{!(C_{11} \otimes \ldots \otimes C_{1n_1}), \ldots, !(C_{m1} \otimes \ldots \otimes C_{mn_m})\}$. An* expansion *of $\mathcal{D}$ is a pair of multisets $\mathcal{D}_1$ and $\mathcal{D}_2$ such that*

$$\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}^L \cup \{C_{11}, \ldots, C_{1n_1}\}^{i_1} \cup \ldots \cup \{C_{m1}, \ldots, C_{mn_m}\}^{i_m}$$

for integers $i_1, \ldots, i_m \geq 0$. A linear expansion *of $\mathcal{D}$ is a pair of multisets $\mathcal{D}_1$ and $\mathcal{D}_2$ such that $\mathcal{D}_1 \cup \mathcal{D}_2 = \mathcal{D}^L$.* $\square$

Note that a linear expansion of $\mathcal{D}$ corresponds to the case in which $i_1 = \ldots = i_m = 0$.

We come now to the definition of resolution proof, Definition 3.7. Such proofs, which are computationally highly deterministic — they are goal-directed, with essentially one left rule — are characterized by uniform linear proofs. Note however that Definition 3.7 provides no mechanism for the non-deterministic decomposition of antecedents and succedents (programs and goals) required by the $\otimes$-rule. A similar issue arises if one considers definite formulae of the form $D_1 \maltese D_2$. We remark that this non-determinism can be handled at the level of the design of an interpreter.

[7]Note that the sequent $p, q, !(p \otimes q) \vdash p \otimes (q \otimes q)$ is not provable in linear sequent calculus.

Definition 3.7 (Resolution Proof) *We define the notion of* resolution proof *by defining a relation* $\vdash_R$. *Let* $\mathcal{D}$ *range over multisets of definite formulae and let* $\cup$ *denote multiset union. A* resolution proof *is a tree regulated by the following rules, which is constructed so that when read from root to leaves, the resolution rules are applied only when no other rule is applicable:*

1. *The axiom judgement is given by:* $\dfrac{}{\mathcal{D} \vdash_R A}$
   where either $\mathcal{D}^L = \{A'\}$ *and* $A = A'[\vec{t}/\vec{x}]$ *or* $\mathcal{D}^L = \emptyset$ *and there exists* $!A' \in \mathcal{D}^I$ *such that* $A = A'[\vec{t}/\vec{x}]$*;*

2. *We shall refer to this rule as the* resolution *rule:*
   $$\frac{\mathcal{D}[\vec{t}/\vec{x}]^g \vdash_R G\,,\mathcal{G} \qquad \{A\} \vdash_R A}{\mathcal{D} \cup \{G' \multimap A'\} \vdash_R A, \mathcal{G}}$$
   where $G \multimap A = (G' \multimap A')[\vec{t}/\vec{x}]$*;*

3. *We shall refer to this rule as the* !-resolution *rule:*
   $$\frac{\mathcal{D}[\vec{t}/\vec{x}]^g \cup [\vec{C}[\vec{t}/\vec{x}]] \cup \{(!(G' \multimap A') \otimes \vec{C})[\vec{t}/\vec{x}]^g\} \vdash_R G\,,\mathcal{G} \qquad \{A\} \vdash_R A}{\mathcal{D} \cup \{!(G' \multimap A') \otimes \vec{C}\} \vdash_R A, \mathcal{G}}$$
   where $G \multimap A = (G' \multimap A')[\vec{t}/\vec{x}]$, *and where* $\vec{C}$ *denotes* $C_1 \otimes \ldots \otimes C_m$ *for* $m \geq 0$*;*

4. *The* $\bot$*-rule:* $$\frac{\mathcal{D} \cup \{A\} \vdash_R \mathcal{G}}{\mathcal{D} \vdash_R A^{\bot}, \mathcal{G}}$$

5. *The* $\otimes$*-rule:* $$\frac{\mathcal{D}^I, \mathcal{D}_1 \vdash_R G_1, \mathcal{G} \qquad \mathcal{D}^I, \mathcal{D}_2 \vdash_R G_2, \mathcal{G}'}{\mathcal{D} \vdash_R G_1 \otimes G_2, \mathcal{G}, \mathcal{G}'}$$
   where $\mathcal{D}_1 \cup \mathcal{D}_2$ *is an expansion of* $\mathcal{D}$*;*

6. *The* $\oplus$*-rule:* $$\frac{\mathcal{D} \vdash_R G_1, \mathcal{G}}{\mathcal{D} \vdash_R G_1 \oplus G_2, \mathcal{G}} \qquad \frac{\mathcal{D} \vdash_R G_2, \mathcal{G}}{\mathcal{D} \vdash_R G_1 \oplus G_2, \mathcal{G}}$$

7. *The* $⅋$*-rule:* $$\frac{\mathcal{D} \vdash_R G_1, G_2, \mathcal{G}}{\mathcal{D} \vdash_R G_1 ⅋ G_2, \mathcal{G}}$$

8. *The* &*-rule:* $$\frac{\mathcal{D}' \vdash_R G_1', \mathcal{G}' \qquad \mathcal{D}'' \vdash_R G_2'', \mathcal{G}''}{\mathcal{D} \vdash_R G_1 \,\&\, G_2, \mathcal{G}}$$
   where $\mathcal{D}'$ *and* $\mathcal{D}''$ *are distinct copies of* D, *and where* $G_1', \mathcal{G}'$ *and* $G_2'', \mathcal{G}''$ *are obtained from* $G_1, \mathcal{G}$ *and* $G_2, \mathcal{G}$ *by the applying the renamings of* $\mathcal{D}'$ *and* $\mathcal{D}''$ *respectively. Note that we maintain "global" and "local" markings;*

9. *The* $\multimap$*-rule:* $$\frac{\mathcal{D} \cup [D] \vdash_R G, \mathcal{G}}{\mathcal{D} \vdash_R D \multimap G, \mathcal{G}}$$

10. *The* $\bigwedge$*-rule:* $$\frac{\mathcal{D} \vdash_R G[y/x], \mathcal{G}}{\mathcal{D} \vdash_R \bigwedge x\,.\,G, \mathcal{G}}$$
    where y *is not free in* $\mathcal{D}$ *or* $\mathcal{G}$*;*

11. *The* $\bigvee$*-rule:* $$\frac{\mathcal{D} \vdash_R G[t/x], \mathcal{G}}{\mathcal{D} \vdash_R \bigvee x\,.\,G, \mathcal{G}}$$

Note that Rules 2 and 3 affect other sequents in the proof, so that a resolution proof is well-defined only if all occurrences of Rules 2 and 3 (and the axioms) yield compatible substitutions. □

Note that the latter rules can be considered to correspond to the right rules of uniform proofs and the former ones to the left rules, in the presence of the decomposition $[-]$.

It should be clear that the !-resolution rule and the second part of the axiom judgement are directly analogous to the corresponding rules in intuitionistic logic. The !-resolution rule clearly has the property that $[\mathcal{P}] \vdash_R A$ if there is a clause $!(G' \multimap A')$ in $[\mathcal{P}]$ such that $G \multimap A = (G' \multimap A')[\vec{t}/\vec{x}]$ and $[\mathcal{P}] \vdash_R G$, which is a more intricate form of the resolution rule given in [12]. Similar remarks apply to the second part of the axiom judgement. For example, consider the sequent

$$\{\, q, !(q \multimap p) \,\} \vdash_R p .$$

By applying the !-resolution rule, we obtain the subgoals

$$\{\, q, !(q \multimap p) \,\} \vdash_R p \text{ and } \{\, p \,\} \vdash_R p.$$

We now proceed to show that resolution proofs behave in the expected manner. First we present a useful lemma.

Lemma 3.8 (Weakening and Contraction) *Let $\mathcal{P}$ be a multiset of mixed clauses, D be a linear clause, $\mathcal{D}$ be a multiset of linear clauses and let $\mathcal{G}$ be a multiset of goal formulae. Then: 1. If $\mathcal{P} \vdash_R \mathcal{G}$, then $\mathcal{P} \cup \{!D\} \vdash_R \mathcal{G}$; 2. If $\mathcal{P} \cup \{!D\} \cup \{!D\} \vdash_R \mathcal{G}$, then $\mathcal{P} \cup \{!D\} \vdash_R \mathcal{G}$; 3. If $\mathcal{P} \cup \{D\}^n \vdash_R \mathcal{G}$ then $\mathcal{P} \cup \{!D\} \vdash_R \mathcal{G}$ for any $n \geq 0$; 4. If $\mathcal{P} \cup \mathcal{D} \vdash_R \mathcal{G}$ then $\mathcal{P} \cup \{! \bigotimes_{C \in [\mathcal{D}]} C\} \vdash_R \mathcal{G}$.* $\square$

One interpretation of this result is that we may replace an occurrence of the resolution rule using the clause D with an occurrence of the !-resolution rule using the clause $!D$. Note that this lemma implies that the set-theoretic properties of formulae commencing with ! in arbitrary linear proofs are preserved in resolution proofs. It is this property which allows us to perform "mixed" intuitionistic and linear deduction in this system. This justifies our remark above that clauses with ! as the principal connective behave in an antecedent in precisely the same way as the clauses of intuitionistic logic programs [12].

We have the following main theorem:

Theorem 3.9 *Let $\mathcal{P}$ be a multiset of definite formulae and let $\mathcal{G}$ be a multiset of goal formulae. Then $\mathcal{P} \vdash \mathcal{G}$ is provable iff $[\mathcal{P}] \vdash_R \mathcal{G}$.* $\square$

In this way we may think of resolution proofs as a relatively efficient way of proving theorems in the given fragment of linear logic. Furthermore, such proofs form a basis for logic programming in this fragment, as the choice of step in the proof-search process is determined by the structure of the goal, and so any (multiset of) goal formulae may be considered as a goal. In particular, this determinism is characterized by resolution proofs.

4 A Sketch of an Interpreter

In this section we sketch the design of an interpreter for linear logic programs. Essentially, the interpreter will execute resolution proofs, however as we noted in Section 4, the definition of resolution proof does not provide a mechanism for calculating the appropriate splitting of the antecedent and succedent in the $\otimes$-rule. This problem is particularly acute in the presence of clauses whose top-level connective is the modality $!$, and so we shall begin with an example of this case. We illustrate our method by presenting a series of examples. We introduce the key notion of *path* in Example 1.

Example 1 Suppose that we are faced with the endsequent

$$!(p \otimes q) \vdash (p \otimes q) \mathbin{\&} (p \otimes (q \otimes q)),$$

which is not provable in linear sequent calculus. We must attempt a resolution proof of

$$\{\, !(p \otimes q) \,\} \vdash_R (p \otimes q) \mathbin{\&} (p \otimes (q \otimes q)),$$

and *fail.* According to the definition of resolution proof, we must identify a suitable expansion of $!(p \otimes q)$ and proceed to identify suitable splittings of that expansion: computationally, this is unacceptable. Our solution is to adopt a *lazy* approach, and to this end we permit the interpreter to construct the following *proto-proof* by modifying the rules of resolution proof:

$$\dfrac{\dfrac{\{\,!(p\otimes q)\,\}\vdash_R p \qquad \{\,!(p\otimes q)\,\}\vdash_R q}{\{\,!(p\otimes q)\,\}\vdash_R p\otimes q}\otimes \qquad \dfrac{\{\,!(p\otimes q)\,\}\vdash^*_R p \qquad \dfrac{\{!(p\otimes q)\}\vdash^*_R q \quad \{!(p\otimes q)\}\vdash^*_R q}{\{\,!(p\otimes q)\,\}\vdash^*_R q\otimes q}\otimes}{\{\,!(p\otimes q)\,\}\vdash^*_R p\otimes(q\otimes q)}\otimes}{\{\,!(p\otimes q)\,\}\vdash^*_R (p\otimes q) \mathbin{\&} (p\otimes(q\otimes q))}\&$$

Define the construction of a *path* in such a proto-proof as follows:

- The endsequent is in every path;
- Traverse the tree towards the leaves, starting at the endsequent:
  - Whenever a &-rule is reached, choose a branch and proceed;
  - Whenever a $\otimes$-rule is reached, proceed along both branches;
- Continue until all branches of the path have reached a leaf.

The proto-proof determines a resolution proof just in case for each possible such path in it, there is an expansion of the antecedent that is compatible with the leaves in the path. For example, the proto-proof given above has a path marked by asterisks (*). For this path, there are 2 occurrences of q in the leaves, so that we need the expansion $\{\, p, p, q, q \,\}$ of $\{\, !(p \otimes q) \,\}$. However, there is only 1 occurrence of p in the leaves of this path, so this expansion is not suitable. We conclude that this proto-proof does not determine a resolution proof of the given endsequent.

Example 2 Now we introduce the resolution rule. Suppose that we are faced with the endsequent

$$p, q, r, (q \otimes r) \multimap p \vdash p \otimes p,$$

which is provable in linear sequent calculus. Since we have that $[p, q, r, (q \otimes r) \multimap p] = \{\, p, q, r, (q \otimes r) \multimap p \,\}$, we must construct a resolution proof of

$$\{\, p, q, r, (q \otimes r) \multimap p \,\} \vdash_R p \otimes p.$$

Adopting our lazy approach we obtain the following proto-proof:

$$\dfrac{\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R p \qquad \dfrac{\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R p \qquad \dfrac{\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R q \quad \{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R r}{\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R q\otimes r}\otimes}{\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R p}\,\text{res}}{\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R p\otimes p}\otimes$$

We construct paths as before (in this case there is just one, marked by *), but upon reaching the lazy resolution rule (labelled by res) we must adjust our calculations. The branch of the lazy version of the resolution rule that corresponds to the branch of the resolution rule of the form $\{A\}\vdash_R A$, in this case $\{p,q,r,(q\otimes r)\multimap p\}\vdash^*_R p$, is not included in the calculation of the usage of the elements of the antecedent; however the resolvant clause, in this case the clause $(q\otimes r)\multimap p$, is used at this point in the proof. We note that for this proto-proof, each of the remaining components, namely p, q, r, is used exactly once in the remaining leaves; and so we conclude that a resolution proof of the given endsequent is indeed determined.

Example 3 Now we introduce the universal quantifier, $\bigwedge$, into the antecedent. Suppose that we are faced with the endsequent

$$\bigwedge x\,.\,(p(x)\otimes q(x))\vdash p(t)\otimes q(u)\,,$$

which is not provable in linear sequent calculus. Since we have that $[\bigwedge x\,.\,(p(x)\otimes q(x))]=\{\,p(x),q(x)\,\}$, with x marked as being global, we must attempt to construct a resolution proof of

$$\{\,p(x),q(x)\,\}\vdash_R p(t)\otimes q(u)\,,$$

and *fail*. Adopting our lazy approach, we obtain the following proto-proof:

$$\dfrac{\{\,p(x),q(x)\,\}\vdash^*_R p(t) \qquad \{\,p(x),q(x)\,\}\vdash^*_R q(u)}{\{\,p(x),q(x)\,\}\vdash^*_R p(t)\otimes q(u)}$$

Each of the leaves (on the unique path marked by *) is of the required form for axioms, but on one branch the global variable x receives the instantiation t and on the other it receives the instantiation u; since these are incompatible, we conclude that the given proto-proof does not determine a resolution proof of the given endsequent.

The examples given above illustrate most of the difficulties encountered in the construction of an interpreter for our notion of linear logic programming. We note that all of the necessary path constructions can be performed dynamically as the search proceeds. However, two further issues are noteworthy.

Firstly, an occurrence of & in a program may result in backtracking, as each formula of the conjunct may be used as the premiss of a proof. However, unlike $\otimes$, only one of the conjuncts is to be used in the proof, and so the conjunct not chosen must not be available for subsequent use. Secondly, a goal in our system is a multiset of formulae, and hence there is the question of which goal to reduce next. Thus we will need a computation rule not only for determining the order of each conjunct in a goal, but also for which goal to reduce next. Clearly a depth-first rule is one possibility; there may be others which are preferable and yet similar in efficiency.

A similar analysis, though somewhat more involved, may be performed for other connectives, including ⅋ and the modality $!$.

Full details of an interpreter are provided in a forthcoming paper by the authors which describes a prototype implementation of linear logic programming.

5 Intuitionistic Logic

In this section we show how proofs in the hereditary Harrop fragment of intuitionistic first-order logic may be encoded as proofs in this fragment of linear logic. In order so to do, we recall some definitions for intuitionistic logic from [12] and [13].

Definition 5.1 (Hereditary Harrop Formulae) *Let A range over atomic formulae. We define the classes of* hereditary Harrop definite formulae *and* hereditary Harrop goal formulae *as follows:*

$$\begin{array}{llll} \textit{Definite formulae} & D & ::= & A \mid D \wedge D \mid G \supset A \mid \forall x \,.\, D \\ \\ \textit{Goal formulae} & G & ::= & A \mid G \vee G \mid G \wedge G \mid D \supset G \\ & & & \mid \forall x \,.\, G \mid \exists x \,.\, G \end{array}$$

An hereditary Harrop program *is a set of closed hereditary Harrop definite formulae. An* hereditary Harrop goal *is a closed hereditary Harrop goal formula.* □ [8]

In [12] a relation $\vdash_o$ denoting *operational provability* is defined and discussed. The reader is referred to [12] and [13] for discussion and proof of:

Proposition 5.2 *Let $\mathcal{P}$ be a hereditary Harrop formula program and let G be a hereditary Harrop goal. Let $\vdash_I$ denote intuitionistic provability. Then $\mathcal{P} \vdash_o G$ iff $\mathcal{P} \vdash_I G$.* □

Below we present an encoding of intuitionistic formulae.

Definition 5.3 (Encoding) *Let D range over hereditary Harrop definite formulae, let $\mathcal{P}$ range over multisets of (linear) definite formulae, let G range over hereditary Harrop goals, and let $\cup$ denote multiset union. We define the linear encoding $(D)^-$ of D and the linear encoding $(G)^+$ of G as follows:*

$$\begin{array}{lll} (A)^+ & =_{\mathrm{def}} & A \\ (G_1 \wedge G_2)^+ & =_{\mathrm{def}} & (G_1)^+ \,\&\, (G_2)^+ \\ (G_1 \vee G_2)^+ & =_{\mathrm{def}} & (G_1)^+ \oplus (G_2)^+ \\ (\exists \vec{x} \,.\, G)^+ & =_{\mathrm{def}} & \bigvee \vec{x} \,.\, (G)^+ \\ (\forall \vec{x} \,.\, G)^+ & =_{\mathrm{def}} & \bigwedge \vec{x} \,.\, (G)^+ \\ (D \supset G)^+ & =_{\mathrm{def}} & (D)^- \multimap (G)^+ \end{array} \qquad \begin{array}{lll} \mathcal{P}^- & =_{\mathrm{def}} & \bigcup_{D \in \mathcal{P}} (D)^- \\ (A)^- & =_{\mathrm{def}} & !A \\ (D_1 \wedge D_2)^- & =_{\mathrm{def}} & !(D_1)^- \otimes !(D_2)^- \\ (\forall \vec{x} \,.\, D)^- & =_{\mathrm{def}} & !(\bigwedge \vec{x} \,.\, (D)^-) \\ (G \supset A)^- & =_{\mathrm{def}} & !((G)^+ \multimap A) \end{array}$$

□

We may think of these encodings as operating at the level of proofs, rather than at the level of formulae. This is due to the way that intuitionistic conjunctions and implications are each translated differently, depending on whether they occur in a positive or negative position. We note that this encoding behaves as expected.

[8] In this case taking $G \supset A$ here is equivalent to taking $G \supset D$. However in the case of linear definite formulae taking $G \supset A$ (Definition 2.3) is not equivalent to taking $G \supset D$. This is because $\multimap$ does not distribute over $\otimes$. Consequently the restriction to $G \supset A$, which is required for the definition of resolution proof, represents a further restriction of the class formulae that we consider. Recall from Section 2 that some simple extensions of this class are possible; for example $G \multimap (A_1 \,\&\, \ldots \,\&\, A_m)$ and $G \multimap \bigwedge \vec{x} \,.\, A$, *etc.*, see [10].

Proposition 5.4 *Let $\mathcal{P}$ be a hereditary Harrop program and let G be a hereditary Harrop goal formula. Then $\mathcal{P} \vdash_o G$ iff $[(\mathcal{P})^-] \vdash_R (G)^+$.* □

Thus we arrive at the following theorem:

Theorem 5.5 *Let $\mathcal{P}$ be a hereditary Harrop program and G be a hereditary Harrop goal. Then the sequent $\mathcal{P} \vdash G$ is provable in intuitionistic logic iff the sequent $(\mathcal{P})^- \vdash (G)^+$ is provable in linear logic.* □

6 Conclusion and Related Work

We have seen how a fragment of linear logic may be used as a logic programming language, and how this fragment may be seen as a conservative extension of an intuitionistic logic programming language.

Related work in linear logic programming is presented in [10], [1]. [10] is the more closely related. Their class of goal formulae is richer than ours in that it permits formulae of the form $!G$. However, this forces their definite formulae to be more constrained. In particular, they cannot have definite formulae of the form $!(D_1 \otimes D_2)$ and so cannot force the use of two clauses an unspecified, but equal, number of times. In fact, they take a notion of linear clause as their starting point, whereas we derive our notion of clause via a proof-theoretic analysis. Indeed, their identification of antecedents with the tensor product of their components, so that the $\otimes$-L rule is not explicitly considered, means that some proof-theoretic subtlety in the analysis of the notion of uniform proof in linear logic is less clear.

7 Acknowledgements

The authors are grateful to Simon Ambler, Gianluigi Bellin, Jean-Yves Girard, Dale Miller, Gordon Plotkin and Lincoln Wallen for their interest in, and encouragement of, this work. Harland is supported a grant of the Australian Research Council through the Machine Intelligence Project. Pym is supported by U.K. Science and Engineering Research Council grant GR/G 58588, "Logical and Semantical Frameworks"; and by European Community ESPRIT Basic Research Action 3245, "Logical Frameworks: Design, Implementation and Experiment".

8 References

[1] Andreoli, J-M., Pareschi, R. *Linear Objects: Logical Processes with Built-in Inheritance.* Proceedings of the International Conference on Logic Programming 496-510, Jerusalem, June, 1990.

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

[3] Asperti, A. *Categorical Topics in Computer Science.* Dottorato Di Ricerca in Informatica, Università di Pisa - Genova - Udine, 1990.

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

[5] Curry, H.B. *The permutability of rules in the classical inferential calculus.* J. Symb. Log. 17, pp. 245-248, 1952.

[6] Gehlot, V., Gunter, C. *Normal Process Representatives.* Proc. 5th Annual IEEE Symposium on Logic in Computer Science. Philadelphia, June 1990. IEEE Computer Society Press.

[7] Gentzen, G. *Untersuchungen über das logische Schliessen.* Math. Zeitschrift 39(1934), pp. 176-210, 405-431.

[8] Girard, J-Y. *Linear Logic.* Theor. Comp. Sci. 50, 1987, pp. 1-102.

[9] Harland, J.A., Pym, D.J. *The Uniform Proof-theoretic Foundation of Linear Logic Programming.* Report ECS-LFCS-90-124. University of Edinburgh, 1990.
[10] Hodas, J., Miller, D. *Logic Programming in a Fragment of Intuitionistic Linear Logic: Extended Abstract.* Proc. 6th Annual IEEE Symposium on Logic in Computer Science. Amsterdam, July 1991. IEEE Computer Society Press.
[11] Kleene, S.C. *Mathematical Logic.* Wiley and Sons, 1968.
[12] Miller, D. *A Logical Analysis of Modules in Logic Programming.* J. Log. Prog. 6(1&2), 1989, pp. 79-108.
[13] Miller, D., Nadathur, G., Pfenning, F., Ščedrov, A. *Uniform Proofs as a Foundation for Logic Programming.* University of Pennsylvania report, MS-CIS-89-36. To appear in the Annals of Pure and Applied Logic.
[14] Pym, D.J., Wallen L.A. *Proof-search in the $\lambda\Pi$-calculus.* In: Logical Frameworks, G. Huet and G. Plotkin (editors), Cambridge University Press, 1991.
[15] Wallen, L.A. *Automated Deduction in Non-classical Logics.* MIT Press, 1989.
[16] Yetter, D. *Quantales and (Non-commutative) Linear Logic.* J. Symb. Log. 55(1), 1990, pp. 41-64.

Appendix A

We present the two-sided linear sequent calculus. We let ϕ and ψ range over formulae and Γ and Δ, *etc.* range over multisets of formulae.

$$\frac{}{\phi \vdash \phi}\ \text{axiom} \qquad\qquad \frac{\Gamma \vdash \phi, \Delta \quad \Gamma', \phi \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \text{cut}$$

$$\frac{\Gamma, \phi, \psi, \Gamma' \vdash \Delta}{\Gamma, \psi, \phi, \Gamma' \vdash \Delta}\ \text{X-L} \qquad\qquad \frac{\Gamma \vdash \Delta, \phi, \psi, \Delta'}{\Gamma \vdash \Delta, \psi, \phi, \Delta'}\ \text{X-R}$$

$$\frac{\Gamma \vdash \phi, \Delta}{\Gamma, \phi^{\perp} \vdash \Delta}\ \perp\text{-L} \qquad\qquad \frac{\Gamma, \phi \vdash \Delta}{\Gamma \vdash \phi^{\perp}, \Delta}\ \perp\text{-R}$$

$$\frac{\Gamma, \phi, \psi \vdash \Delta}{\Gamma, \phi \otimes \psi \vdash \Delta}\ \otimes\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi, \Delta \quad \Gamma' \vdash \psi, \Delta'}{\Gamma, \Gamma' \vdash \phi \otimes \psi, \Delta, \Delta'}\ \otimes\text{-R}$$

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, \phi \mathbin{\&} \psi \vdash \Delta} \quad \frac{\Gamma, \psi \vdash \Delta}{\Gamma, \phi \mathbin{\&} \psi \vdash \Delta}\ \&\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi, \Delta \quad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \mathbin{\&} \psi, \Delta}\ \&\text{-R}$$

$$\frac{\Gamma, \phi \vdash \Delta \quad \Gamma, \psi \vdash \Delta}{\Gamma, \phi \oplus \psi \vdash \Delta}\ \oplus\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash \phi \oplus \psi, \Delta} \quad \frac{\Gamma \vdash \psi, \Delta}{\Gamma \vdash \phi \oplus \psi, \Delta}\ \oplus\text{-R}$$

$$\frac{\Gamma, \phi \vdash \Delta \quad \Gamma', \psi \vdash \Delta'}{\Gamma, \Gamma', \phi \maltese \psi \vdash \Delta, \Delta'}\ \maltese\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi, \psi, \Delta}{\Gamma \vdash \phi \maltese \psi, \Delta}\ \maltese\text{-R}$$

$$\frac{\Gamma \vdash \phi, \Delta \quad \Gamma', \psi \vdash \Delta'}{\Gamma, \Gamma', \phi \multimap \psi \vdash \Delta, \Delta'}\ \multimap\text{-L} \qquad\qquad \frac{\Gamma, \phi \vdash \psi, \Delta}{\Gamma \vdash \phi \multimap \psi, \Delta}\ \multimap\text{-R}$$

$$\frac{\Gamma, \phi \vdash \Delta}{\Gamma, !\phi \vdash \Delta}\ !\text{-L} \qquad\qquad \frac{!\Gamma \vdash \phi, ?\Delta}{!\Gamma \vdash !\phi, ?\Delta}\ !\text{-R}$$

$$\frac{!\Gamma, \phi \vdash ?\Delta}{!\Gamma, ?\phi \vdash ?\Delta}\ ?\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi, \Delta}{\Gamma \vdash ?\phi, \Delta}\ ?\text{-R}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, !\phi \vdash \Delta}\ W!\text{-L} \qquad\qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash ?\phi, \Delta}\ W?\text{-R}$$

$$\frac{\Gamma, !\phi, !\phi \vdash \Delta}{\Gamma, !\phi \vdash \Delta}\ C!\text{-L} \qquad\qquad \frac{\Gamma \vdash ?\phi, ?\phi, \Delta}{\Gamma \vdash ?\phi, \Delta}\ C?\text{-R}$$

$$\frac{\Gamma, \phi[t/x] \vdash \Delta}{\Gamma, \bigwedge x . \phi \vdash \Delta}\ \bigwedge\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi[y/x], \Delta}{\Gamma \vdash \bigwedge x . \phi, \Delta}\ \bigwedge\text{-R}$$

$$\frac{\Gamma, \phi[y/x] \vdash \Delta}{\Gamma, \bigvee x . \phi \vdash \Delta}\ \bigvee\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi[t/x], \Delta}{\Gamma \vdash \bigvee x . \phi, \Delta}\ \bigvee\text{-R}$$

where x is not free in Γ, Δ.