On Goal-Directed Proofs in Multiple-Conclusioned Intuitionistic Logic

James Harland, Tatjana Lutovac, Michael Winikoff
Department of Computer Science
RMIT, GPO Box 2476V
Melbourne, 3001
{jah,tanja,winikoff}@cs.rmit.edu.au
http://www.cs.rmit.edu.au/~{jah,tanja,winikoff}

Abstract

A key property in the definition of logic programming languages is the completeness of goal-directed proofs. This concept originated in the study of logic programming languages for intuitionistic logic in the (single-conclusioned) sequent calculus LJ, but has subsequently been adapted to multiple-conclusioned systems such as those for linear logic. Given these developments, it seems interesting to investigate the notion of goal-directed proofs for a multiple-conclusioned sequent calculus for intuitionistic logic, in that this is a logic for which there are both single-conclusioned and multiple-conclusioned systems (although the latter are less well known than the former). In this paper we show that the language obtained for the multiple-conclusioned system differs from that for the single-conclusioned case, and discuss the consequences of this result.

Keywords: Multiple-conclusioned intuitionistic logic, goal-directed proofs, logic programming languages, hereditary Harrop formulae, proof search.

1 Introduction

Logic programming is based upon the observation that if certain restrictions are placed on the class of formulae that can be used, then statements of mathematical logic can be interpreted as computer programs. In particular, computation consists of a search for a proof of a given goal from a given program, and the restrictions placed on the formulae ensure that this proof search is sufficiently deterministic. The best known such restriction is to allow programs to consist of *Horn clauses* and goals to consist of existentially quantified conjunctions of atoms [12], which form the basis of the language *Prolog* [23].

Whilst such languages have been used in a wide number of applications [22], it is not immediately clear why any restriction on the class of formulae is necessary; in particular, what is it that distinguishes a logic programming language from a theorem prover? Whilst a general answer to this question remains elusive, there have been various logic programming languages based on other classes of formulae than Horn clauses, and indeed on logics other than classical logic. These include extensions to Horn clauses such as allowing implications, universal quantifiers, and negations in the bodies of clauses [3, 15], the incorporation of higher-order facilities [15], and negations and disjunctions in the heads of clauses [16], as well as languages based on on linear logic (including Lygon[9], Forum[14], LinLog [1], LO [2], Lolli [10], ACL [11], and $\mathcal{LC}$ [25]).

Many of these languages are based on the notion of a *goal-directed proof* [15], which, roughly speaking, requires that the goal be decomposed before the program, and hence the computation uses the program as a context, but the goal as the controlling sequence of instructions. This idea was first presented in the context of intuitionistic logic, and has in many of the above cases been generalised to multiple-conclusioned logics such as classical logic and linear logic (although, it must be said, there appear to be at least two distinct such generalisations). However, in its original form [15], the notion of a goal-directed proof is derived directly from Gentzen's sequent calculus for intuitionistic logic known as LJ. This calculus may be seen as a special case of the calculus LK for classical logic, in that LJ is obtained from LK by restricting the succedents to contain at most one formula (thus LJ is often referred to as *single-conclusioned*). This property is then exploited in the notion of a goal-directed proof by requiring that the outermost connective of the formula in the succedent (if any) is to be reduced before any of the connectives which appear in the antecedent. This approach is not without its problems, as discussed in [27], but to date it has been the most successful approach to identifying logic programming languages in single-conclusioned systems. In particular, this notion of proof has lead to the study of the class of formulae known as *hereditary Harrop formulae*, which may be used as the basis of both first-order and higher-order logic programming languages [15]. There is some evidence that this class of formulae is, in some sense, maximal [7] (at least for the first-order case).

Thus it would seem that the identification of logic programming languages in intuitionistic logic is a solved problem. However, it is less widely known that there are multiple-conclusioned sequent calculi for intuitionistic logic [26]. Whilst these are not as well known as LJ, they have been of some interest for the relationship between intuitionistic and classical inference [20, 21]. Given such inference systems, the question naturally arises as to what logic programming languages would look like in such systems, and what the results of the previous analysis would be. This is a particularly interesting question given that there has been a significant amount of investigation of notions of goal-directed provability for multiple-conclusioned systems such as linear logic [1, 10, 19] and classical logic [8, 17, 18]. Thus it seems appropriate to investigate the design of logic programming languages via goal-directed provability for a multiple-conclusioned sequent calculus for intuitionistic logic.

In this paper we investigate such a sequent calculus from the point of view of goal-directed proofs. In particular, we investigate the completeness properties of this class of proofs, which are fundamental to the identification of logic programming languages. We then compare our results to those obtained in the single-conclusioned case.

2 Preliminaries

2.1 Sequent Calculi

Sequent calculi are due originally to Gentzen [5] and are often used in the analysis of proof systems. A sequent $\Gamma \vdash \Delta$ may be thought of as stating that if *all* the formulae in Γ are true, then *at least one* of the formulae in Δ is true. Γ is referred to as the *antecedent* and Δ as the *succedent*.

The sequent calculus for classical logic, LK, is the best known (and arguably the simplest). The rules for this calculus are given below.

$$\frac{}{F \vdash F}\ \mathrm{Axiom} \qquad \frac{\Gamma \vdash F, \Delta \quad \Gamma, F \vdash \Delta}{\Gamma \vdash \Delta}\ \mathrm{Cut} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta}\ \mathrm{WL} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash F, \Delta}\ \mathrm{WR}$$

$$\frac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta}\,\mathrm{CL} \qquad \frac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta}\,\mathrm{CR} \qquad \frac{\Gamma, F_1, F_2, \Gamma' \vdash \Delta}{\Gamma, F_2, F_1, \Gamma' \vdash \Delta}\,\mathrm{IL} \qquad \frac{\Gamma \vdash \Delta, F_1, F_2, \Delta'}{\Gamma \vdash \Delta, F_2, F_1, \Delta'}\,\mathrm{IR}$$

$$\frac{\Gamma, F_1, F_2 \vdash \Delta}{\Gamma, F_1 \wedge F_2 \vdash \Delta}\,\wedge\mathrm{L} \qquad \frac{\Gamma \vdash F_1, \Delta \quad \Gamma \vdash F_2, \Delta}{\Gamma \vdash F_1 \wedge F_2, \Delta}\,\wedge\mathrm{R} \qquad \frac{\Gamma, F_1 \vdash \Delta \quad \Gamma, F_2 \vdash \Delta}{\Gamma, F_1 \vee F_2 \vdash \Delta}\,\vee\mathrm{L} \qquad \frac{\Gamma \vdash F_1, F_2, \Delta}{\Gamma \vdash F_1 \vee F_2, \Delta}\,\vee\mathrm{R}$$

$$\frac{\Gamma \vdash F_1, \Delta \quad \Gamma, F_2 \vdash \Delta}{\Gamma, F_1 \to F_2 \vdash \Delta} \to \mathrm{L} \qquad \frac{\Gamma, F_1 \vdash F_2, \Delta}{\Gamma \vdash F_1 \to F_2, \Delta} \to \mathrm{R} \qquad \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta}\,\neg\mathrm{L} \qquad \frac{\Gamma, F \vdash \Delta}{\Gamma \vdash \neg F, \Delta}\,\neg\mathrm{R}$$

$$\frac{\Gamma, F[t/x] \vdash \Delta}{\Gamma, \forall x F \vdash \Delta}\,\forall\mathrm{L} \qquad \frac{\Gamma \vdash F[y/x], \Delta}{\Gamma \vdash \forall x F, \Delta}\,\forall\mathrm{R} \qquad \frac{\Gamma, F[y/x] \vdash \Delta}{\Gamma, \exists x F \vdash \Delta}\,\exists\mathrm{L} \qquad \frac{\Gamma \vdash F[t/x], \Delta}{\Gamma \vdash \exists x F, \Delta}\,\exists\mathrm{R}$$

The rules $\exists$L and $\forall$R have the usual side condition that y is not free in Γ, Δ or F.

LK has the cut-elimination property [5], i.e. that any proof containing occurrences of the Cut rule can be replaced with a (potentially much larger) proof in which there are no occurrences of the Cut rule.

It is well known that the sequent calculus contains redundancies, in that there may be several trivially different proofs of the same sequent. In particular, the order of the rules can often be permuted, in that given a sequence of inference rules, we can change the order of the rules to obtain an equivalent sequence (i.e. one which has the same root and leaves as the original).

In order to study such properties, we require some further terminology [4]. The *active formulae* of an inference are the formulae which are present in the premise(s), but not in the conclusion. The *principal* formula of an inference is the formula which is present in the conclusion, but not in the premise(s). Intuitively, the inference converts the active formulae into the principal formula (but as discussed in [13], this is sometimes too simplistic).

When looking to permute the order of two inferences, it is necessary to check that the principal formula of the upper inference is not an active formula of the lower one; otherwise, no permutation is possible. When this property occurs, the two inferences are said to be in *permutation position* [4, 13].

For example, consider the two inferences below.

$$\frac{\dfrac{\dfrac{q \vdash p, q, r}{q \vdash p, q \vee r}\,\vee\mathrm{R}}{\neg p, q \vdash q \vee r}\,\neg\mathrm{L}}{\neg p \wedge q \vdash q \vee r}\,\wedge\mathrm{L} \qquad \frac{\dfrac{\dfrac{q \vdash p, q, r}{\neg p, q \vdash q, r}\,\neg\mathrm{L}}{\neg p, q \vdash q \vee r}\,\vee\mathrm{R}}{\neg p \wedge q \vdash q \vee r}\,\wedge\mathrm{L}$$

In either inference, we have the following:

Rule	Principal Formula	Active Formulae
$\wedge$L	$\neg p \wedge q$	$\neg p$, q
$\neg$L	$\neg p$	p
$\vee$R	$q \vee r$	q, r

Note that in the left-hand inference, as $\neg p$ is both the principal formula of $\neg$L and an active formula of $\wedge$L, $\neg$L and $\wedge$L are not in permutation position. On the other hand, as the active formula of $\neg$L is p, this is distinct from the principal formula of $\vee$R, which is $q \vee r$, and hence $\neg$L and $\vee$R are in permutation position. In particular, we can permute $\vee$R below $\neg$L (or alternatively $\neg$L above $\vee$R) resulting in the right-hand inference above.

2.2 Intuitionistic Logic and LJ

The standard sequent calculus for intuitionistic logic, LJ, can be obtained from LK by requiring that in every sequent $\Gamma \vdash \Delta$ the succedent Δ contains at most one formula. This has the effect of dropping the rules CR and IR, and restricts the WR rule to the case in which the succedent of the premise must be empty. Similar remarks apply to the $\neg$L and $\neg$R rules. The only other rules with any changes of significance are the $\rightarrow$L and $\vee$R rules, which have the following form in LJ:

$$\frac{\Gamma \vdash F_1 \quad \Gamma, F_2 \vdash \Delta}{\Gamma, F_1 \rightarrow F_2 \vdash \Delta} \rightarrow \mathrm{L} \qquad \frac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \vee F_2} \vee\mathrm{R}$$

The $\rightarrow$L rule thus omits the duplication of Δ in the left-hand premise.

The $\vee$R rule must choose which of F_1 and F_2 is to appear in the premise. In fact, in the presence of the WR and CR rules, this rule is equivalent to the one given for LK above. As we shall see, this is a crucial difference between LJ and the multiple-conclusioned version.

LJ also has the cut-elimination property [5].

2.3 Multiple-Conclusioned Systems for Intuitionistic Logic

Below is the multiple-conclusioned system taken from [26].

$$\frac{}{F \vdash F} \text{ Axiom} \qquad \frac{\Gamma \vdash \Delta}{\Gamma, F \vdash \Delta} \text{ WL} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash F, \Delta} \text{ WR}$$

$$\frac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta} \text{ CL} \qquad \frac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta} \text{ CR} \qquad \frac{\Gamma, F_1, F_2, \Gamma' \vdash \Delta}{\Gamma, F_2, F_1, \Gamma' \vdash \Delta} \text{ IL} \qquad \frac{\Gamma \vdash \Delta, F_1, F_2, \Delta'}{\Gamma \vdash \Delta, F_2, F_1, \Delta'} \text{ IR}$$

$$\frac{\Gamma, F_1, F_2 \vdash \Delta}{\Gamma, F_1 \wedge F_2 \vdash \Delta} \wedge\mathrm{L} \qquad \frac{\Gamma \vdash F_1, \Delta \quad \Gamma \vdash F_2, \Delta}{\Gamma \vdash F_1 \wedge F_2, \Delta} \wedge\mathrm{R} \qquad \frac{\Gamma, F_1 \vdash \Delta \quad \Gamma, F_2 \vdash \Delta}{\Gamma, F_1 \vee F_2 \vdash \Delta} \vee\mathrm{L} \qquad \frac{\Gamma \vdash F_1, F_2, \Delta}{\Gamma \vdash F_1 \vee F_2, \Delta} \vee\mathrm{R}$$

$$\frac{\Gamma, F[y/x] \vdash \Delta}{\Gamma, \exists x F \vdash \Delta} \exists\mathrm{L} \qquad \frac{\Gamma \vdash F[t/x], \Delta}{\Gamma \vdash \exists x F, \Delta} \exists\mathrm{R} \qquad \frac{\Gamma, F[t/x] \vdash \Delta}{\Gamma, \forall x F \vdash \Delta} \forall\mathrm{L} \qquad \frac{\Gamma \vdash F[y/x]}{\Gamma \vdash \forall x F, \Delta} \forall\mathrm{R}$$

$$\frac{\Gamma \vdash F_1, \Delta \quad \Gamma, F_2 \vdash \Delta}{\Gamma, F_1 \rightarrow F_2 \vdash \Delta} \rightarrow \mathrm{L} \qquad \frac{\Gamma, F_1 \vdash F_2}{\Gamma \vdash F_1 \rightarrow F_2, \Delta} \rightarrow \mathrm{R} \qquad \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta} \neg\mathrm{L} \qquad \frac{\Gamma, F \vdash}{\Gamma \vdash \neg F, \Delta} \neg\mathrm{R}$$

The rules $\exists$L and $\forall$R have the usual side condition that y is not free in Γ, Δ or F.

Following [20, 21], we refer to this system as LM. Unlike LJ, contraction on the right may be used arbitrarily here. Note also that this is effectively negated in some instances by the form of the rules for $\forall$R, $\rightarrow$R and $\neg$R.

Note also that the $\vee$R rule is classical (ie the LK rule), and the rules $\forall$R, $\rightarrow$R and $\neg$R are different from both LK and LJ. Following Wallen [26], let us call these latter rules *special* rules.

As an illustration of the differences between LK, LJ and LM, consider Pierce's formula $((p \rightarrow q) \rightarrow p) \rightarrow p$, which is provable classically, but not intuitionistically. The LK proof is below, as are the corresponding failed attempts in LJ and LM respectively (in left to right order).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash q, p}\,\text{Ax}}{\vdash p \to q, p}\,\to\text{R} \qquad \dfrac{}{p \vdash p}\,\text{Ax}}{(p \to q) \to p \vdash p}\,\to\text{L}}{\vdash ((p \to q) \to p) \to p}\,\to\text{R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{X}{p \vdash q}}{\vdash p \to q}\,\to\text{R} \qquad \dfrac{}{p \vdash p}\,\text{Ax}}{(p \to q) \to p \vdash p}\,\to\text{L}}{\vdash ((p \to q) \to p) \to p}\,\to\text{R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{X}{p \vdash q}}{\vdash p \to q, p}\,\to\text{R} \qquad \dfrac{}{p \vdash p}\,\text{Ax}}{(p \to q) \to p \vdash p}\,\to\text{L}}{\vdash ((p \to q) \to p) \to p}\,\to\text{R}$$

Note that in LK, the $\to$L rule does not make a choice between $p \to q$ and p, whereas in LJ it chooses p. In LM, the $\to$L does not make a choice between $p \to q$ and p, but the $\to$R rule does.

3 Goal-Directed Proofs

The logic programming interpretation of a sequent $\Gamma \vdash \Delta$ is that the antecedent Γ represents the program, and the succedent Δ the goal. Hence when searching for a proof of $\Gamma \vdash \Delta$ (i.e. performing computation), the search should be "driven" by Δ (and thus be goal-directed). The proof-theoretic characterisation of this property is the notion of *uniform proof.*

Definition 1 *An LJ proof is* uniform *if for every sequent $\Gamma \vdash \Delta$ in which Δ is a non-atomic formula, the inference rule used to derive $\Gamma \vdash \Delta$ is the right rule for the principal connective of Δ.*

Thus the search process must reduce a non-atomic succedent before it looks at the program. We also need a proof-theoretic account of resolution, which is given by the notion of a *simple proof* [15].

Definition 2 *An LJ proof is* simple *if for every occurrence of $\to$L the right hand premise is an axiom.*

Clearly it is possible for an LJ proof to be neither uniform nor simple. Hence the question is to identify a class of formulae for which simple uniform proofs are complete (i.e. do not "miss" any consequences).

The fragment known as *hereditary Harrop formulae* has these properties and is defined as follows: Let A range over atomic formulae.

$$\begin{array}{lll} \textit{Definite formulae} \quad D & ::= & A \mid D \wedge D \mid G \to A \mid \forall x \,.\, D \\[1em] \textit{Goal formulae} \quad G & ::= & A \mid G \vee G \mid G \wedge G \mid D \to G \mid \forall x \,.\, G \mid \exists x \,.\, G \end{array}$$

A program, then, is a set of definite formulae, and a goal is a goal formula. We then have the following theorem.

Theorem 1 (Miller et. al [15]) *Let $P \vdash G$ be a hereditary Harrop sequent. Then $P \vdash_I G$ iff $P \vdash G$ has a simple uniform proof in LJ.*

4 LP Languages in LM

We now turn to the problem of identifying logic programming languages in LM. Following the above pattern, we need to find an appropriate conception of goal-directed proof in LM. Having done so, a class of formulae is then a logic programming language if for any P and G in the appropriate class, $P \vdash G$ is provable iff $P \vdash G$ has a goal-directed proof. We can establish such a result by considering *permutabilities* of rules and showing that a given proof of $P \vdash G$ can always be transformed using permutabilities into a goal-directed proof.

Now clearly the only rules in LM which differ from LK are ∀R, →R and ¬R, and hence these are the only ones whose permutation behaviour will differ from LK. Note that we are particularly interested in permuting right rules downwards (i.e. towards the root of the proof). The following table summarises when we can permute a right rule above a left to a left above a right.

	∀R	¬R	→R	∃R	∧R	∨R
∨L	no	no	no	yes	yes	yes
→L (right)	no	no	no	yes	yes	yes
→L (left)	*can be*	*eliminated*	*using W*	yes	yes	yes
∧L	yes	yes	yes	yes	yes	yes
∀L	yes	yes	yes	yes	yes	yes
∃L	yes	yes	yes	no	yes	yes
¬L	yes	yes	yes	yes	yes	yes

Note that we distinguish between the right rule appearing in permutation position above the left premise of →L and above the right premise. We do not do this for ∨L as the two are symmetric. For the →L (left) case, note the following transformation:

$$\dfrac{\dfrac{\vdots \atop \Gamma, F_3 \vdash F_4}{\Gamma \vdash F_1, F_3 \to F_4, \Delta}\to \mathrm{R} \qquad {\vdots \atop \Gamma, F_2 \vdash F_3 \to F_4, \Delta}}{{\Gamma, F_1 \to F_2 \vdash F_3 \to F_4, \Delta \atop \vdots}}\to \mathrm{L} \qquad \Longrightarrow \qquad \dfrac{\dfrac{\vdots \atop \Gamma, F_3 \vdash F_4}{\Gamma \vdash F_3 \to F_4, \Delta}\to \mathrm{R}}{{\Gamma, F_1 \to F_2 \vdash F_3 \to F_4, \Delta \atop \vdots}}WL$$

A point to note is that in LM, the ∨R rule can be permuted below the ∨L rule, which is not the case in LJ (but is the case in LK). As a result, it is possible to use disjunctions positively in programs in LM. This gives a proof-theoretic characterisation of the notion of *disjunctive logic programs* [16], which have been used to model certain types of uncertain information. This may be thought of as a particular instance of the general observation that there is a trade-off between the expressivity of the language and the strength of the properties of the search strategy. In this case, no choice has to be made when the ∨R rule is applied, and hence a larger fragment of the logic may be used; *quid pro quo*, the resulting proofs no longer have the disjunctive property (i.e. that if $F_1 \vee F_2$ is provable, then so is F_i for some $i = 1, 2$). Note, though, that the ∃R rule cannot be permuted below the ∃L rule (just as in LK), and hence there is no corresponding property for ∃.

Hence the main observation is that the special rules do not permute downwards past ∨L or →L on the right. Below is a proof in which →R occurs above the right premise of →L, but cannot be permuted downwards.

$$\dfrac{\dfrac{p \vdash p, q \to r, q \qquad q \vdash p, q \to r, q}{p \vee q \vdash p, q \to r, q}\vee\mathrm{L} \qquad \dfrac{p \vee q, q, r \vdash r}{p \vee q, r \vdash q \to r, q}\to \mathrm{R}}{p \vee q, p \to r \vdash q \to r, q}\to \mathrm{L}$$

Attempting to prove this sequent with →R results in an unprovable premise, as below.

$$\dfrac{\dfrac{\dfrac{X}{p \lor q, q \vdash p, r} \qquad \dfrac{}{p \lor q, r, q \vdash r}\ \mathrm{Ax}}{p \lor q, p \to r, q \vdash r}\ {\to}\,\mathrm{L}}{p \lor q, p \to r \vdash q \to r, q}\ {\to}\,\mathrm{R}$$

Hence, in order for a class of formulae to be a logic programming language, we need to ensure that the “no” cases in the above table cannot occur.

We have two orthogonal choices:

1. $\exists$L versus $\exists$R
2. The special rules ($\forall$R, $\neg$R and $\to$R) versus $\lor$L and $\to$L.

Note that the rules $\forall$L, $\land$L, $\neg$L, $\lor$R and $\land$R can be freely used. This yields the following four combinations:

1. Right rules: $\land, \lor, \forall, \neg, \to$, Left rules: $\forall, \land, \neg, \exists$.
2. Right rules: $\land, \lor, \forall, \neg, \to, \exists$, Left rules: $\forall, \land, \neg$.
3. Right rules: $\land, \lor$, Left rules: $\forall, \exists, \land, \lor, \neg, \to$
4. Right rules: $\land, \lor, \exists$, Left rules: $\forall, \land, \lor, \neg, \to$

The first two possibilities don’t appear to be very useful since they do not allow implication on the left and thus don’t even allow Horn clauses.

Of the remaining two possibilities, the latter is more promising, as the former does not include existential quantifiers in goals, and hence also does not generalise Horn clauses. Hence the most useful language is the last one, which, when compared to hereditary Harrop formulae allows disjunctions and negations on the left, but disallows universal quantifiers and implications on the right.

One question that quickly arises in any discussion of goal-directedness in a multiple-conclusioned setting is that there is now a choice to be made between applicable right rules (whereas in the single-conclusioned case there is only one). Clearly there are only two possibilities: either the choice is arbitrary (and hence any choice will suffice) or it is not (and so more care must be taken to maintain completeness). The former is what is assumed in Forum, and hence all right rules must permute over each other. The latter is what is assumed in Lygon, and hence the possible execution strategies must be derived from an analysis of the permutation properties of the right rules, which in the case of Lygon, is based around Andreoli’s analysis of such rules [1].

The following table summarises permutability properties among the right rules. A *yes* indicates that the right rule of the column can be permuted below the right rule of the row. A *no* indicates that this is not possible and a *N/A* indicates that it is not possible for the pair of rules to occur in permutation position. A *no?* indicates that although a normal permutation is not possible a sub-proof with the same premises and conclusion is possible which only applies the special rule. For example consider the transformation below:

$$\begin{array}{c}\vdots\\ \dfrac{\dfrac{\Gamma, G \vdash H}{\Gamma \vdash F_1, F_2, G \to H}\ {\to}\,\mathrm{R}}{\Gamma \vdash F_1 \lor F_2, G \to H}\ \lor\mathrm{R}\\ \vdots\end{array} \quad \Longrightarrow \quad \begin{array}{c}\vdots\\ \dfrac{\Gamma, G \vdash H}{\Gamma \vdash F_1 \lor F_2, G \to H}\ {\to}\,\mathrm{R}\\ \vdots\end{array}$$

	$\forall$-R	$\neg$-R	$\rightarrow$-R	$\exists$-R	$\wedge$-R	$\vee$-R
$\forall$-R	N/A	N/A	N/A	N/A	N/A	N/A
$\neg$-R	N/A	N/A	N/A	N/A	N/A	N/A
$\rightarrow$-R	N/A	N/A	N/A	N/A	N/A	N/A
$\exists$-R	no?	no?	no?	yes	yes	yes
$\wedge$-R	no?	no?	no?	yes	yes	yes
$\vee$-R	no?	no?	no?	yes	yes	yes

Hence we arrive at the following notion of goal-directness and class of formulae.

Definition 3 *An LM proof is* uniform *if every sequent which contains a non-atomic formula in the succedent is the conclusion of a right rule.*

Definition 4 *LM-definite formulae and LM-goal formulae are given by the grammar:*

$$\textit{LM-definite formulae} \quad D \quad ::= \quad A \mid D \wedge D \mid D \vee D \mid \neg D \mid G \rightarrow D \mid \forall x \,.\, D$$

$$\textit{LM-goal formulae} \quad G \quad ::= \quad A \mid G \vee G \mid G \wedge G \mid \exists x \,.\, G$$

It is then straightforward to show the following result.

Proposition 1 *Let $\mathcal{P}$ be a set of LM-definite formulae and $\mathcal{G}$ be a set of LM-goal formulae. Then $\mathcal{P} \vdash \mathcal{G}$ has an LM-proof iff $\mathcal{P} \vdash \mathcal{G}$ has a uniform proof.*

As for simple proofs, we need to restrict the formulae in the programs to be *clausal*, i.e. of the form $G \rightarrow A$ rather than $G \rightarrow D$. This is done so that when permuting other left rules down below $\rightarrow$L on the right, we can be sure that the two rules are always in permutation position (as an atom can never be the principal formula). Hence we arrive at the following definition.

Definition 5 *Clausal LM-definite formulae are given by the grammar:*

$$\textit{Clausal LM-definite formulae} \quad D \quad ::= \quad A \mid D \wedge D \mid D \vee D \mid \neg D \mid G \rightarrow A \mid \forall x \,.\, D$$

Then it is straightforward to show the following result from the permutation properties by a simple inductive argument.

Proposition 2 *Let $\mathcal{P}$ be a set of clausal LM-definite formulae and $\mathcal{G}$ be a set of LM-goal formulae. Then $\mathcal{P} \vdash \mathcal{G}$ has an LM-proof iff $\mathcal{P} \vdash \mathcal{G}$ has a simple uniform proof.*

In the LJ case, there is no increase in power by allowing clauses of the form $G \rightarrow D$, due to the following intuitionistic equivalences:

$$G \rightarrow (D_1 \wedge D_2) \equiv (G \rightarrow D_1) \wedge (G \rightarrow D_2)$$
$$G \rightarrow (G' \rightarrow D) \equiv (G \wedge G') \rightarrow D$$
$$G \rightarrow (\forall x D) \equiv \forall x (G \rightarrow D) \text{ where } x \text{ is not free in } G.$$

In the LM case, this is no longer true, due to the presence of clauses of the form $D_1 \vee D_2$, and that $G \rightarrow (D_1 \vee D_2)$ is not intuitionistically equivalent to $(G \rightarrow D_1) \vee (G \rightarrow D_2)$. However, it should be noted that this equivalence does hold in a slightly stronger logic, in which this is one of the *Independence of Premise* rules. Further discussion on this point is beyond the scope of this paper; interested readers are referred to [6, 24].

5 Conclusions and Further Work

We have seen that the permutation properties of LM mean that the straightforward application of the notion of goal-directed proof from the single-conclusioned case results in a different class of formulae than hereditary Harrop formulae. This suggests that a further and more subtle analysis is needed. The language discussed here is basically a version of Horn clauses which may contain negations and disjunctions. Hence this class of formulae (and proofs in LM) may be useful for investigations of programs which may contain such connectives, but it is not at all clear what, if any, relationship it has to hereditary Harrop formulae.

Another topic of interest is the relationship between search in LM and search in LJ. In particular, the search properties of LM may be thought of as allowing "delayed" choices when compared with LJ, particularly for the $\vee$R and $\rightarrow$L rules as mentioned above. This means that an attempt at a proof in LM may correspond to more than one such attempt in LJ; a correspondence of this sort seems worthy of further investigation.

Another relevant direction is the slightly stronger logic in which the Independence of Premise rules hold. As we have seen, this is relevant to issues of clausal decomposition, as well as to operational equivalences of programs [6]. An investigation of the proof theory of such a logic and its relation to LM would be particularly interesting.

6 Acknowledgements

The authors are grateful to David Pym for stimulating discussions. The first author is grateful for the hospitality of the Department of Computer Science of Queen Mary and Westfield College, University of London during a period of sabbatical leave.

References

[1] J.-M. Andreoli. Logic Programming with Focusing Proofs in Linear Logic. *Journal of Logic and Computation*, 2(3), 1992.

[2] J.-M. Andreoli and R. Pareschi. Linear Objects: Logical Processes with Built-in Inheritance. In David H. D. Warren and Péter Szeredi, editors, *Proceedings of the Seventh International Conference on Logic Programming*, pages 496–510, Jerusalem, 1990. The MIT Press.

[3] K. Clark. Negation as Failure. In H. Gallaie and J. Minker, editors, *Logic and Databases*, pages 293–323. Plenum Press, 1978.

[4] D. Galmiche and G.Perrier. On Proof Normalisation in Linear Logic. *Theoretical Computer Science*, 135:67–110, 1994.

[5] G. Gentzen. Untersuchungen über das logische Schliessen. *Math. Zeit.*, 39:176–210,405–431, 1934.

[6] J. Harland. On Normal Forms and Equivalence for Logic Programs. In Krzysztof Apt, editor, *Proceedings of thed Joint International Conference and Symposium on Logic Programming*, pages 146–160, Washington, DC, 1992. ALP, MIT Press.

[7] J. Harland. A Proof-Theoretic Analysis of Goal-Directed Provability. *Journal of Logic and Computation*, 4(1):69–88, January 1994.

[8] J. Harland. On Goal-Directed Provability in Classical Logic. *Computer Languages*, 23:161–178, 1997.

[9] J. Harland, D. Pym, and M. Winikoff. Programming in Lygon: An Overview. In M. Wirsing, editor, *Lecture Notes in Computer Science*, pages 391–405. Springer, July 1996.

[10] J. Hodas and D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic. *Information and Computation*, 110(2):327–365, 1994.

[11] N. Kobayash and A. Yonezawa. ACL - A Concurrent Linear Logic Programming Paradigm. In Dale Miller, editor, *Logic Programming - Proceedings of the 1993 International Symposium*, pages 279–294, Vancouver, Canada, 1993. The MIT Press.

[12] R. Kowalski. Predicate Logic as a Programming Language. In *Information Processing 74*, Amsterdam, 1974. North-Holland.

[13] T. Lutovac and J. Harland. Towards the Automation of the Design of Logic Programming Languages. Technical Report 97-30, Department of Computer Science, RMIT, 1997.

[14] D. Miller. Forum: A Multiple-Conclusion Specification Logic. *Theoretical Computer Science*, 165(1):201–232, 1996.

[15] D. Miller, G. Nadathur, F. Pfenning, and A. Šč̌edrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[16] J. Minker and A. Rajasekar. A Fixpoint Semantics for Disjunctive Logic Programs. *Journal of Logic Programming*, 9(1):45–74, July 1990.

[17] G. Nadathur. Uniform Provability in Classical Logic. *Journal of Logic and Computation*, 8(2):209–230, 1998.

[18] G. Nadathur. Correspondences between Classical, Intuitionistic and Uniform Provability. to appear in *Theoretical Computer Science*, 1999.

[19] D. Pym and J. Harland. A Uniform Proof-theoretic Investigation of Linear Logic Programming. *Journal of Logic and Computation*, 4(2):175–207, April 1994.

[20] E. Ritter, D. Pym, and L. Wallen. On the Intuitionistic Force of Classical Search. to appear in *Theoretical Computer Science*, 1999.

[21] E. Ritter, D. Pym, and L. Wallen. Proof Terms for Classical and Intuitionistic Resolution. to appear in the *Journal of Logic and Computation*, 1999.

[22] Leon Sterling, editor. *The Practice of Prolog*. MIT Press, 1990.

[23] Leon Sterling and Ehud Shapiro. *The Art of Prolog*. MIT Press, 1994.

[24] A. Troelstra. *Lectures on Linear Logic*. CSLI Lecture Notes No. 29, 1992.

[25] P. Volpe. Concurrent Logic Programming as Uniform Linear Proofs. In G. Levi and M. Rodríguez-Artalejo, editors, *Proceedings of the Conference on Algebraic and Logic Programming*, pages 133–149. Springer, 1994.

[26] Lincoln Wallen. *Automated Deduction in Nonclassical Logics*. MIT Press, 1990.

[27] M. Winikoff. *Logic Programming with Linear Logic*. PhD thesis, University of Melbourne, 1997.