Goal-Directed Proof Search in Multiple-Conclusioned Intuitionistic Logic

James Harland Tatjana Lutovac Michael Winikoff

Department of Computer Science
Royal Melbourne Institute of Technology
GPO Box 2476V, Melbourne, 3001, AUSTRALIA
{*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). In this paper we show that the language obtained for the multiple-conclusioned system differs from that for the single-conclusioned case, show how hereditary Harrop formulae can be recovered, and investigate contraction-free fragments of the logic.

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 searching for a proof of a goal from a program, and the restrictions placed on both the program and the goal 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, which form the basis of the language *Prolog*.

When looking to extend such results to other logical systems, we proceed by starting from a logic $\mathcal{L}$ (or indeed a set of inference rules for $\mathcal{L}$) a proof search strategy $\mathcal{S}$, and then determining a set of goal formulae $\mathcal{G}$ and a set of program formulae $\mathcal{P}$ such that $\mathcal{S}$ is *complete* with respect to $\mathcal{L}$. This process amounts to a systematic method for designing logic programming languages, and has two important properties. Firstly, such a systematic process can uncover a richer, more expressive programming language. For example, analysis in intuitionistic logic has uncovered extensions to Horn clauses such as allowing implications, universal quantifiers, and negations in the bodies of clauses [3, 16], incorporation of higher-order facilities [16], and negations and disjunctions in the heads of clauses [17]. Secondly, this process can be applied to logics other than classical (or intuitionistic) logic. For example, a number of logic programming languages have been derived from linear logic including Lygon[11], Forum[15], LinLog [1], LO [2], Lolli [12], ACL [13], and $\mathcal{L}C$ [21].

A popular proof search strategy is the notion of *goal-directed proof* [16], which, roughly speaking, requires that the goal be decomposed before the program, and hence the computation uses the program as a context, and the goal as the controlling sequence of instructions. The original presentation of goal-directed proof (and its formalisation – *uniformity*) were derived for the *single-conclusioned* sequent calculus LJ. However, this notion has been generalised to multiple-conclusioned sequent calculi in much of the work on linear logic[1]. As pointed out in [23, section 3.1], there are some weaknesses in this approach, but it remains the best-known way to derive logic programming languages from inference rules.

In this paper we focus on the generalisation of uniformity to multiple-conclusioned logics. We investigate this question in the familiar territory of intuitionistic logic. The standard sequent calculus for intuitionistic logic is LJ, which is single-conclusioned. It is known that *hereditary Harrop formulae* are a logic programming language in intuitionistic logic, using goal-directed proof search in LJ. Further, there is evidence that this class of formulae is, in some sense, maximal [9] (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 [22]. Whilst these are not as well known as LJ, they have been of some interest for the relationship between intuitionistic and classical inference [20]. 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, 15, 19, 21] and classical logic [10, 18]. Thus it seems appropriate to investigate the design of logic programming languages via goal-directed provability for a multiple-conclusioned system for intuitionistic logic, and to compare the results with the single-conclusioned case.

2 Preliminaries

2.1 Sequent Calculi

Sequent calculi are due originally to Gentzen [7] and are often used in the analysis of proof systems. This is because sequent calculus rules are local (and hence conceptually straightforward to implement) and there is a natural distinction between programs and goals.

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). Below we give a few of the rules for this calculus.

[1] Actually, there appear to be at least two distinct such generalisations.

$$\frac{}{F \vdash F}\ \text{Axiom} \qquad \frac{\Gamma \vdash F, \Delta \quad \Gamma, F \vdash \Delta}{\Gamma \vdash \Delta}\ \text{Cut} \qquad \frac{\Gamma \vdash F_1, F_2, \Delta}{\Gamma \vdash F_1 \vee F_2, \Delta}\ \vee\text{R} \qquad \frac{\Gamma \vdash F, \Delta}{\Gamma, \neg F \vdash \Delta}\ \text{L}^{\perp}$$

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

LK has the cut-elimination property [7], 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. Both of the other sequent calculus systems used in this paper (LJ and LM) also have the cut-elimination property.

2.2 Permutabilities

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 [6]. 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 [14], 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* [6, 14].

For example, consider the two inferences below.

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

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.

2.3 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. Amongst other changes, this means that the conclusion of the $\neg$L rule must have an empty succedent. Other rules which are significantly affected are the $\to$L and $\vee$R rules, which have the following form in LJ:

$$\dfrac{\Gamma \vdash F_1 \quad \Gamma, F_2 \vdash F}{\Gamma, F_1 \to F_2 \vdash F} \to \mathrm{L} \qquad \dfrac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \vee F_2} \vee\mathrm{R}$$

The $\to$L rule thus omits the duplication of F in the left-hand premise and the $\vee$R rule must choose which of F_1 and F_2 is to appear in the premise. As we shall see, this is a crucial difference between LJ and the multiple-conclusioned version.

2.4 Multiple-Conclusioned Systems for Intuitionistic Logic

Below is the multiple-conclusioned system taken from [22].

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

$$\dfrac{\Gamma, F, F \vdash \Delta}{\Gamma, F \vdash \Delta}\ \mathrm{CL} \qquad \dfrac{\Gamma \vdash F, F, \Delta}{\Gamma \vdash F, \Delta}\ \mathrm{CR} \qquad \dfrac{\Gamma, F_1, F_2 \vdash \Delta}{\Gamma, F_1 \wedge F_2 \vdash \Delta}\ \wedge\mathrm{L} \qquad \dfrac{\Gamma \vdash F_1, \Delta \quad \Gamma \vdash F_2, \Delta}{\Gamma \vdash F_1 \wedge F_2, \Delta}\ \wedge\mathrm{R}$$

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

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

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

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

Following [20], we refer to this system as **LM**. Unlike LJ, contraction on the right may be used arbitrarily here. Note that this is effectively negated by the form of the rules for $\forall$R, $\to$R and $\neg$R. Note also that the $\vee$R rule is classical (ie the LK rule), and the rules $\forall$R, $\to$R and $\neg$R are different from both LK and LJ. Following Wallen [22], let us call these latter rules *special* rules.

As an illustration of the differences between LK, LJ and LM, consider Peirce's formula $((p \to q) \to p) \to p$, 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).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash q, p}\ \mathrm{Ax}}{\vdash p \to q, p} \to \mathrm{R} \quad \dfrac{}{p \vdash p}\ \mathrm{Ax}}{(p \to q) \to p \vdash p} \to \mathrm{L}}{\vdash ((p \to q) \to p) \to p} \to \mathrm{R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{✗}{p \vdash q}}{\vdash p \to q} \to \mathrm{R} \quad \dfrac{}{p \vdash p}\ \mathrm{Ax}}{(p \to q) \to p \vdash p} \to \mathrm{L}}{\vdash ((p \to q) \to p) \to p} \to \mathrm{R} \qquad \dfrac{\dfrac{\dfrac{\dfrac{✗}{p \vdash q}}{\vdash p \to q, p} \to \mathrm{R} \quad \dfrac{}{p \vdash p}\ \mathrm{Ax}}{(p \to q) \to p \vdash p} \to \mathrm{L}}{\vdash ((p \to q) \to p) \to p} \to \mathrm{R}$$

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

2.5 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* [16].

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* [16].

Definition 2. *An LJ proof is* simple *if for every occurrence of $\rightarrow$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* (HHF) has these properties and is defined as follows, where A ranges over atomic formulae.

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

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 [16]).
Let $P \vdash G$ be a hereditary Harrop sequent. Then $P \vdash G$ iff $P \vdash G$ has a simple uniform proof in LJ.

3 Deriving a Logic Programming Language 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 in LM. 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. We are particularly interested in permuting right rules downwards (i.e. towards the root of the proof). A non-permuting right rule indicates a situation that needs to be avoided; this implies a constraint on the class of formulae considered to be a logic programming language.

3.1 Permutation Properties of LM

Since the only rules in LM which differ from LK are ∀R, →R and ¬R these are the only ones whose permutation behaviour will differ from LK. Note that in LK all of the propositional logical rules permute with each other. 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	✗	✗	✗	✓	✓	✓
→L (right)	✗	✗	✗	✓	✓	✓
→L (left)	*can be*	*eliminated*	*using W*	✓	✓	✓
∧L	✓	✓	✓	✓	✓	✓
∀L	✓	✓	✓	✓	✓	✓
∃L	✓	✓	✓	✗	✓	✓
¬L	✓	✓	✓	✓	✓	✓

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 the row marked "can be eliminated ..." corresponds to the transformation which replaces the left inference below with the right one, thus eliminating altogether the occurrence of the →L rule.

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

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* [17], 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 expressiveness 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 $\vee$L or $\rightarrow$L on the right. Below is a proof in which $\rightarrow$R occurs above the right premise of $\rightarrow$L, but cannot be permuted downwards.

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

Attempting to prove this sequent with $\rightarrow$R results in an unprovable premise:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p, q \vdash r, p} \quad \dfrac{✗}{q, q \vdash r, p}}{p \vee q, q \vdash r, p}\ \vee\mathrm{L} \qquad \dfrac{}{p \vee q, r, q \vdash r}}{p \vee q, p \rightarrow r, q \vdash r}\ \rightarrow \mathrm{L}}{p \vee q, p \rightarrow r \vdash q \rightarrow r, q}\ \rightarrow \mathrm{R}$$

3.2 Identifying a Subset of the Logic

In order for a class of formulae to be a logic programming language (using a goal-directed proof search) we need to ensure that the no ("✗") cases in the above table cannot occur. We do this by ensuring that one of the non-permuting rules cannot occur by constraining the class of formulae to limit occurrences of the relevant connective. We have two orthogonal choices: (1) $\exists$L versus $\exists$R; and (2) The special rules ($\forall$R, $\neg$R and $\rightarrow$R) versus $\vee$L and $\rightarrow$L.

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

	Right rules	Left rules
1.	$\wedge, \vee, \forall, \neg, \rightarrow$	$\forall, \wedge, \neg, \exists$
2.	$\wedge, \vee, \forall, \neg, \rightarrow, \exists$	$\forall, \wedge, \neg$
3.	$\wedge, \vee$	$\forall, \exists, \wedge, \vee, \neg, \rightarrow$
4.	$\wedge, \vee, \exists$	$\forall, \wedge, \vee, \neg, \rightarrow$

The first three possibilities don't appear to be very useful since they do not include Horn clauses: the first two possibilities do not allow implication on the left (and hence do not allow rules in programs) and the third possibility does not allow existential quantification in goals.

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 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 "✓" indicates that the right rule of the column can be permuted below the right rule of the row. A "-" indicates that it is not possible for the pair of rules to occur in permutation position and a "(✗)" 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:

$$\dfrac{\dfrac{\Gamma, G \vdash H}{\Gamma \vdash F_1, F_2, G \to H}\to \mathrm{R}}{\Gamma \vdash F_1 \vee F_2, G \to H}\vee\mathrm{R} \quad \Longrightarrow \quad \dfrac{\Gamma, G \vdash H}{\Gamma \vdash F_1 \vee F_2, G \to H}\to \mathrm{R}$$

This is similar to the case involving a special rule above the left premise of $\to$L.

	$\forall$-R	$\neg$-R	$\to$-R	$\exists$-R	$\wedge$-R	$\vee$-R
$\forall$-R	-	-	-	-	-	-
$\neg$-R	-	-	-	-	-	-
$\to$-R	-	-	-	-	-	-
$\exists$-R	(✗)	(✗)	(✗)	✓	✓	✓
$\wedge$-R	(✗)	(✗)	(✗)	✓	✓	✓
$\vee$-R	(✗)	(✗)	(✗)	✓	✓	✓

Thus, for LM, all right rules effectively permute over each other and so proof search can arbitrarily choose an applicable right rule without any loss of completeness.

3.3 Formal Results

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:*

LM-definite formulae $D ::= A \mid D \wedge D \mid D \vee D \mid \neg G \mid G \to D \mid \forall x \,.\, D$
LM-goal formulae $G ::= A \mid G \vee G \mid G \wedge G \mid \exists x \,.\, G$

Note that although negations occurring positively in programs are permitted according to uniformity, they are in some sense not goal directed in that it is possible to have programs which are provable regardless of the goal given. For example, $p, \neg p \vdash G$ is provable for any goal formula G.

Theorem 2 (Uniformity). *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.*

Proof. (sketch) The basic idea is that since all of the possible right rules permute down over all of the possible left rules we can eliminate non-uniform inferences by permuting right rules down so they are beneath left rules. The details are more subtle and are omitted due to space limitations. □

As for simple proofs, we need to restrict the formulae in the programs to be *clausal*, i.e. of the form $G \to A$ rather than $G \to D$. This is done so that when permuting other left rules down below $\to$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:*

$$D ::= A \mid D \wedge D \mid D \vee D \mid G \to A \mid \forall x \,.\, D$$

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

Theorem 3. *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.*

Proof. By theorem 2, there is a uniform proof of $\mathcal{P} \vdash \mathcal{G}$. From the permutation properties we know that any occurrences of $\exists$R, $\vee$R or $\wedge$R can be permuted down past $\to$L on the right. By a similar argument, it can be shown that any occurrences of $\wedge$L, $\neg$L, $\vee$L and $\forall$L can be permuted downwards. Hence the right premise of an occurrence of $\to$L must be the conclusion of either an axiom, or another $\to$L. The following permutation can be applied to the highest non-simple occurrence of $\to$L in a chain of $\to$L to make it simple. We repeat this until all occurrences of $\to$L in the chain are simple.

$$\dfrac{\Gamma, G_2 \to A_2 \vdash G_1, \Delta \qquad \dfrac{\Gamma, A_1 \vdash G_2, \Delta \qquad \overline{\Gamma, A_1, A_2 \vdash \Delta}}{\Gamma, G_2 \to A_2, A_1 \vdash \Delta} \to \mathrm{L}}{\Gamma, G_1 \to A_1, G_2 \to A_2 \vdash \Delta} \to \mathrm{L}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, G_2 \to A_2 \vdash G_1, \Delta}{\Gamma, G_2 \to A_2 \vdash G_1, G_2, \Delta} \mathrm{WR} \qquad \dfrac{\Gamma, A_1 \vdash G_2, \Delta}{\Gamma, G_2 \to A_2, A_1 \vdash G_2, \Delta} \mathrm{WL}}{\Gamma, G_2 \to A_2, G_1 \to A_1 \vdash G_2, \Delta} \to \mathrm{L} \qquad \Gamma, G_1 \to A_1, G_2 \to A_2, A_2 \vdash \Delta}{\Gamma, G_2 \to A_2, G_1 \to A_1, G_2 \to A_2 \vdash \Delta} \to \mathrm{L}}{\Gamma, G_1 \to A_1, G_2 \to A_2 \vdash \Delta} \mathrm{CL}$$

Consider the sequent $\Gamma, A_1, A_2 \vdash \Delta$ in the first inference. Since it is the conclusion of an axiom rule (modulo structural rules) we have that either $A_2 \vdash \Delta$, or $A_1 \vdash \Delta$, or $F \vdash \Delta$ for some $F \in \Delta$. In the last two cases we can simply delete the top occurrence of $\to$L altogether thus making the remaining $\to$L simple. In the remaining case we have that $\Gamma, G_1 \to A_1, G_2 \to A_2, A_2 \vdash \Delta$ and hence the permutation has produced a simple occurrence of $\to$L as desired. □

Theorem 4. *LM-definite and LM-goal formulae are not a logic programming language in LJ.*

Proof. LM-definite and LM-goal formulae allow disjunction as a top-level connective. The sequent $p \vee q \vdash p \vee q$ is provable, but does not have a uniform (in the sense of definition 1) proof. □

Hence the analyses for LM and LJ lead to different logic programming languages. However, whilst HHF do not qualify "directly" as a logic programming language in LM, it is possible to recover HHF as a logic programming language in LM by a simple analysis of the role of disjunctions in programs.

4 Definite Formulae in LM

One interpretation of the results of the previous sections is that whilst it is possible to use disjunctions positively in programs in an intuitionistic setting by using LM rather than LJ, the cost is that the richer language of HHF cannot be used. On the other hand, it is well-known that in LJ, HHF can be used, but at the cost of not allowing disjunctions in programs. However, we can recover HHF without changing inference rules by omitting disjunctions in programs (and hence not using the ∨L in proofs). We explore this issue in this section.

It is not hard to show that in LM, WL can always be permuted upwards, and WR can either be permuted upwards or "absorbed" by the special rules. For the latter case, note the transformation below.

$$\dfrac{\dfrac{\Gamma \vdash F[y/x]}{\Gamma \vdash \forall x F, \Delta}\ \forall R}{\Gamma \vdash \forall x F, F', \Delta}\ WR \qquad \Longrightarrow \qquad \dfrac{\Gamma \vdash F[y/x]}{\Gamma \vdash \forall x F, F', \Delta}\ \forall R$$

Hence we can use the form of the axiom rule below, and omit the WL and WR rules.

$$\overline{\Gamma, F \vdash F, \Delta}\ Axiom'$$

This means that we can show the following result.

Theorem 5. *Let $\Gamma \vdash \Delta$ be a provable sequent in LM in which ∨L does not occur. Then either $\Gamma \vdash$ is provable in LM or $\Gamma \vdash F$ for some $F \in \Delta$ is provable in LM.*

Note the strength of the contrapositive of this result; if a proof requires multiple conclusions, then it must include an occurrence of ∨L.

Proof. We proceed by induction on the size of the proof. In the base case, $\Gamma \vdash \Delta$ is just an axiom, and clearly the result is trivially true. Hence we assume the result is true for all proofs of no more than a given size. The cases for CL, ∧L, ∧R, ∃L, ∃R, ∀L, ¬L and the special rules are all trivial. That leaves CR, ∨R, and →L.

CR: The previous sequent is $\Gamma \vdash F, F, \Delta$, and so by the hypothesis we have that either $\Gamma \vdash F'$ for some $F' \in \Delta$, in which case we are done, or $\Gamma \vdash F$ in which case we are done.

∨**R:** The previous sequent is $\Gamma \vdash F_1, F_2, \Delta$, and so by the hypothesis we have that either $\Gamma \vdash F$ for some $F \in \Delta$, in which case we are done, or $\Gamma \vdash F_i$, in which case we can derive $\Gamma \vdash F_1 \vee F_2$ via WR and ∨R.

→**L:** The previous sequents are $\Gamma \vdash F_1, \Delta$ and $\Gamma, F_2 \vdash \Delta$, and so by the hypothesis, we have that $\Gamma, F_2 \vdash F'$ for some $F' \in \Delta$ and either $\Gamma \vdash F$ for some $F \in \Delta$, in which case we can derive $\Gamma, F_1 \to F_2 \vdash F$ by WL, or we have $\Gamma \vdash F_1$. In the latter case we have $\Gamma \vdash F_1$ and $\Gamma, F_2 \vdash F'$, and hence we have $\Gamma, F_1 \to F_2 \vdash F'$.

□

Thus once (positive) disjunctions are removed from programs (i.e. antecedents), we essentially recover LJ.

In particular, this shows that we can use the following versions of →L and ∨R:

$$\frac{\Gamma \vdash F_1 \quad \Gamma, F_2 \vdash F}{\Gamma, F_1 \to F_2 \vdash F} \qquad \frac{\Gamma \vdash F_i}{\Gamma \vdash F_1 \vee F_2}$$

These are clearly just the LJ rules. This means that by using this form of →L, we can recover the downwards permutability of the special rules over →L. For example, consider the →R rule and the transformation below:

$$\dfrac{\Gamma \vdash F_1 \quad \dfrac{\Gamma, F_2, F_3 \vdash F_4}{\Gamma, F_2 \vdash F_3 \to F_4, \Delta} \to R}{\Gamma, F_1 \to F_2 \vdash F_3 \to F_4, \Delta} \to L \quad \Longrightarrow \quad \dfrac{\dfrac{\dfrac{\Gamma \vdash F_1}{\Gamma, F_3 \vdash F_1} WL \quad \Gamma, F_2, F_3 \vdash F_4}{\Gamma, F_1 \to F_2, F_3 \vdash F_4} \to L}{\Gamma, F_1 \to F_2 \vdash F_3 \to F_4, \Delta} \to R$$

Thus we can recover the completeness of uniform and simple (LJ) proofs by using particular properties of LM. This, together with the earlier arguments about disjunctions in LM, may be seen as evidence that the approach using LM is more general than using LJ.

5 Contraction-free fragments

One issue which becomes relevant in the analysis involving LM is the role of contraction in succedents. In LJ, such contraction is forbidden; in LM, a naive interpretation would require that each formula be copied before being used as the principal formula of a rule application. However, it is not hard to show that the CR rule is only necessary when such a rule application is ∃R — in all other cases the CR rule can either be permuted upwards or can be eliminated.

That such contractions are necessary is shown by the following proof.

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{p(a) \vdash p(a), p(b)} \quad \overline{p(b) \vdash p(a), p(b)}}{p(a) \vee p(b) \vdash p(a), p(b)} \vee L}{p(a) \vee p(b) \vdash p(a), \exists x p(x)} \exists R}{p(a) \vee p(b) \vdash \exists x p(x), \exists x p(x)} \exists R}{p(a) \vee p(b) \vdash \exists x p(x)} CR$$

Hence an implementation will require existentially quantified goals to be copied. However, Theorem 5 shows that such copying will not be necessary in the absence of disjunctions in programs, or for any fragment which does not contain existentially quantified goals. In particular, this argument shows that the propositional fragment does not require contraction on the right.

It is then interesting to pursue the question of whether contraction on the left is required in the propositional case. Dyckhoff [4] has shown that it is possible to use a more intricate proof system in which contraction is not needed at all for any propositional fragment. In our case, we are interested in determining whether the standard rules of LM are contraction-free for various propositional fragments.

An intriguing result is that propositional Horn clauses are contraction-free, but propositional hereditary Harrop formulae are not. That the latter are not may be shown by the following proof:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overline{p \vdash p}}{p \vdash p \lor (p \to q)}\ \lor R}{(p \lor (p \to q)) \to q, p \vdash q}\ \to L}{(p \lor (p \to q)) \to q \vdash p \to q}\ \to R}{(p \lor (p \to q)) \to q \vdash p \lor (p \to q)}\ \lor R}{(p \lor (p \to q)) \to q, (p \lor (p \to q)) \to q \vdash q}\ \to L}{(p \lor (p \to q)) \to q \vdash q}\ CL
$$

However, if we try to omit the contraction, we quickly arrive at an unprovable sequent:

$$
\dfrac{\dfrac{✗}{\vdash p \lor (p \to q)}}{(p \lor (p \to q)) \to q \vdash q}\ \to L
$$

It is interesting to note that Dyckhoff shows in [4] that the formula $\neg\neg(p \lor \neg p)$ requires contraction in LJ; this formula is essentially the same as the above formula under the transformation of $\neg F$ to $F \to \bot$.

We now proceed to show that propositional Horn clauses do not require contraction. We denote by $\mathcal{D}_{\land,\lor,\to,\neg}\mathcal{G}_{\land,\lor,\to,\neg}$ the fragment defined by the following rules:
$D ::= A \mid D \land D \mid D \lor D \mid \neg G \mid G \to A$
$G ::= A \mid G \land G \mid G \lor G \mid \neg D \mid D \to G$

We use similar notation for smaller fragments such as $\mathcal{D}_{\land,\to}\mathcal{G}_{\land,\lor,\to}$.

Definition 6. *Given an occurrence $\Gamma, F \vdash F, \Delta$ of the $Axiom'$ rule, we refer to Γ as the* context.

In a proof Φ of a propositional sequent $\Gamma \vdash \Delta$, a formula F in Γ is passive *if is not the principal formula of any rule occurrence in Φ, and F is in the context of every occurrence of the $Axiom'$ rule.*

Now in order to show that a given fragment is contraction-free, we proceed by showing that if a contraction is used, then the formula that is copied by the rule is passive (i.e. plays no active part in the proof).

It is not hard to show that CL permutes up past all propositional rules except →L.

Hence, we need only consider occurrences of CL which are immediately below →L, and for which the principal formula of →L is an active formula of CL, i.e.

$$
\dfrac{\dfrac{\Gamma, F_1 \to F_2 \vdash F_1}{\Gamma, F_1 \to F_2, F_1 \to F_2 \vdash F_2}\ \to L}{\Gamma, F_1 \to F_2 \vdash F_2}\ CL
$$

Theorem 6. *Let Φ be a proof of a $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\wedge,\vee}$ sequent $P \vdash G$. Then for every occurrence*

$$\dfrac{\dfrac{\dfrac{\Psi}{\Gamma, F_1 \rightarrow F_2 \vdash F_1}}{\Gamma, F_1 \rightarrow F_2, F_1 \rightarrow F_2 \vdash F_2} \rightarrow L}{\Gamma, F_1 \rightarrow F_2 \vdash F_2} \, CL$$

of CL and →L in Φ, $F_1 \rightarrow F_2$ is passive in Ψ.

Proof. We proceed by induction on the number of occurrences of →L in Φ. Consider an occurrence of →L closest to the leaves. As there are no occurrences of →L in Ψ, $F_1 \rightarrow F_2$ is clearly passive.

Hence we assume that the result holds for proofs which contain no more than a given number of occurrences of →L.

Consider an occurrence of →L as above. If $F_1 \rightarrow F_2$ is not passive in Ψ, then there must be an occurrence of →L in Ψ in which $F_1 \rightarrow F_2$ is the principal formula. It is not hard to see that as the goals consist only of conjunctions and disjunctions (and hence the antecedents in the proof can never be changed), this must be of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Xi}{\Gamma, F_1 \rightarrow F_2 \vdash F_1}}{\Gamma, F_1 \rightarrow F_2, F_1 \rightarrow F_2 \vdash F_2} \rightarrow L}{\Gamma, F_1 \rightarrow F_2 \vdash F_2} \, CL}{\Psi}}{\Gamma, F_1 \rightarrow F_2 \vdash F_1}}{\Gamma, F_1 \rightarrow F_2, F_1 \rightarrow F_2 \vdash F_2} \rightarrow L}{\Gamma, F_1 \rightarrow F_2 \vdash F_2} \, CL$$

By the hypothesis we know that $F_1 \rightarrow F_2$ is passive in Ξ.

Now as there is a sequent identical to one closer to the leaves, we can eliminate the part of the proof between these two occurrences, and in particular we can remove the copy of the identical sequent closer to the root. □

In a similar manner, it is not hard to show that the $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\wedge,\rightarrow,\neg}$ fragment is also contraction-free.

Theorem 7. *Let Φ be a proof of a $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\wedge,\rightarrow,\neg}$ sequent $P \vdash G$. Then for every occurrence*

$$\dfrac{\dfrac{\dfrac{\Psi}{\Gamma, F_1 \rightarrow F_2 \vdash F_1}}{\Gamma, F_1 \rightarrow F_2, F_1 \rightarrow F_2 \vdash F_2} \rightarrow L}{\Gamma, F_1 \rightarrow F_2 \vdash F_2} \, CL$$

of CL and →L in Φ, $F_1 \rightarrow F_2$ is passive in Ψ.

Proof. Similar to the above argument. □

Hence we arrive at the following classification:

- $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\wedge,\vee}$ (Horn clauses) is contraction-free.
- $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\wedge,\rightarrow,\neg}$ is contraction-free.

– $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\vee,\rightarrow}$ is not contraction-free (see above).
– $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\vee,\neg}$ is not contraction-free (Dyckhoff [4]).

Note that the third case implies that propositional hereditary Harrop formulae are not contraction-free. Note also that these cases cover all the fragments of $\mathcal{D}_{\wedge,\rightarrow}\mathcal{G}_{\wedge,\vee,\rightarrow,\neg}$.

6 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, and in particular that disjunctions may be used in programs. We have also seen that it is possible to recover hereditary Harrop formulae by not using such disjunctions. This suggests that LM is a potentially more general framework for logic programming languages based on intuitionistic logic than LJ.

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 property of interest is the precise notion of equivalence used. It is known that for hereditary Harrop formulae in LJ, there is no increase in power by allowing clauses of the form $G \rightarrow D$, due to the following intuitionistic equivalences[2]:

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

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 (called Gödel-Dummett logic in [5]), in which this is one of the *Independence of Premise* rules. This logic is also relevant to issues of program equivalence [8], and so an investigation of the proof theory of such a logic and its relation to LM would be particularly interesting.

7 Acknowledgements

The authors are grateful to Pablo Armelin, Roy Dyckhoff and 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.

[2] One consequence of the first equivalence is that there is no benefit to considering multi-headed clauses for hereditary Harrop formulae as in Lygon, Forum, or LO.

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. R. Dyckhoff. Contraction-free Sequent Calculi for Intuitionistic Logic. *Journal of Symbolic Logic*, 57:795–807, 1992.
5. R. Dyckhoff. A Deterministic Terminating Sequent Calculus for Gödel-Dummett logic. *Logic Journal of the IGPL*, 7:319–326, 1999.
6. D. Galmiche and G.Perrier. On Proof Normalisation in Linear Logic. *Theoretical Computer Science*, 135:67–110, 1994.
7. G. Gentzen. Untersuchungen über das logische Schliessen. *Math. Zeit.*, 39:176–210,405–431, 1934.
8. J. Harland. On Normal Forms and Equivalence for Logic Programs. In Krzysztof Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 146–160, Washington, DC, 1992. ALP, MIT Press.
9. J. Harland. A Proof-Theoretic Analysis of Goal-Directed Provability. *Journal of Logic and Computation*, 4(1):69–88, January 1994.
10. J. Harland. On Goal-Directed Provability in Classical Logic. *Computer Languages*, 23:161–178, 1997.
11. 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.
12. J. Hodas and D. Miller. Logic Programming in a Fragment of Intuitionistic Linear Logic. *Information and Computation*, 110(2):327–365, 1994.
13. 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.
14. 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.
15. D. Miller. Forum: A Multiple-Conclusion Specification Logic. *Theoretical Computer Science*, 165(1):201–232, 1996.
16. D. Miller, G. Nadathur, F. Pfenning, and A. Ščedrov. Uniform Proofs as a Foundation for Logic Programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.
17. J. Minker and A. Rajasekar. A Fixpoint Semantics for Disjunctive Logic Programs. *Journal of Logic Programming*, 9(1):45–74, July 1990.
18. G. Nadathur. Uniform Provability in Classical Logic. *Journal of Logic and Computation*, 8(2):209–230, 1998.
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. 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.
22. Lincoln Wallen. *Automated Deduction in Nonclassical Logics*. MIT Press, 1990.
23. M. Winikoff. *Logic Programming with Linear Logic*. PhD thesis, University of Melbourne, 1997.