ON RESOLUTION IN FRAGMENTS OF CLASSICAL LINEAR LOGIC (EXTENDED ABSTRACT)*

J.A. Harland† — University of Melbourne, Australia

D.J. Pym‡ — University of Edinburgh, Scotland, U.K.

Abstract

We present a proof-theoretic foundation for logic programming in Girard's linear logic. We exploit the permutability properties of two-sided linear sequent calculus to identify appropriate notions of uniform proof, definite formula, goal formula, clause and resolution proof for fragments of linear logic. The analysis of this paper extends earlier work by the present authors to include negative occurrences of $\mathbin{⅋}$ (par) and positive occurrences of ! (of course !) and ? (why not ?). These connectives introduce considerable difficulty. We consider briefly some of the issues related to the mechanical implementation of our resolution proofs.

1 Introduction

An interesting recent development in logic of some significance for theoretical computer science is *linear logic* [3], [4], a relevance logic lacking the both structural rules of weakening and contraction, except via the exponential operators ! (of course !) and ? (why not ?).

We discuss the naturally-arising question of the significance of linear logic for logic programming. In particular, we investigate the definition of logic programming languages within linear logic and consider the characterization of their novel features.

The two-sided linear sequent calculus is given in Appendix A. We let ϕ, ψ, χ and ξ range over (linear) formulae, and let Γ, Γ' *etc.* range over antecedents and Δ, Δ' *etc.* range over succedents. We work with cut-free linear sequent calculus throughout. We consider antecedents and succedents to be multisets of formulae and so are able to suppress all occurrences of the *exchange* rules in linear proofs. We assume that the reader is familiar with treatment of a logic program and a goal as the antecedent and succedent of a certain sequent.

We take as our point of conceptual departure the *uniform proofs* of Miller *et al.* [10], [9], and the class of hereditary Harrop formulae of intuitionistic logic. In particular, we assume that the characteristic feature of logic programming is goal-directed proof-search. More precisely, there is a search operation corresponding to each logical connective, and when searching for a proof of a given goal one applies the search operation that corresponds to the outermost connective of that goal, and then to the outermost connective of each subgoal so generated, *etc.*. Furthermore, we assume that it must be possible to rewrite the program so that just one left rule, an appropriate notion of *resolution*[1] rule is required.

In linear logic, the identification of a class of computationally appealing proofs, uniform proofs, is somewhat more intricate than in logics that have been considered previously. For example, in first-order hereditary Harrop formulae [10], [9], the search operation corresponding to each connective that can occur in a goal is given by the right rule for that connective. Moreover, it is easy to show that the strategy of constructing proofs (from root to leaves) by applying a right rule wherever one is applicable is complete. This property arises from the permutability properties of the rules of the intuitionistic sequent calculus. For example, it should be clear that an intuitionistic subproof of the form

*Proceedings of the Russian Conference on Logic Programming and Automated Reasoning 30-41, St. Petersburg, July, 1992. Published as Lecture Notes in Artificial Intelligence 624, Springer-Verlag, 1992.

†Supported by a grant of the Australian Research Council through the Machine Intelligence Project.

‡Supported by U.K. Science and Engineering Research Council grant GR/G 58588, "Logical and Semantical Frameworks".

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

$$\cfrac{\cfrac{\Gamma, \chi, \xi \vdash \phi \qquad \Gamma, \chi, \xi \vdash \psi}{\Gamma, \chi, \xi \vdash \phi \wedge \psi}\ \wedge\text{-R}}{\Gamma, \chi \wedge \xi \vdash \phi \wedge \psi}\ \wedge\text{-L}$$

can be replaced by one of the form

$$\cfrac{\cfrac{\Gamma, \chi, \xi \vdash \phi}{\Gamma, \chi \wedge \xi \vdash \phi}\ \wedge\text{-L} \qquad \cfrac{\Gamma, \chi, \xi \vdash \psi}{\Gamma, \chi \wedge \xi \vdash \psi}\ \wedge\text{-L}}{\Gamma, \chi \wedge \xi \vdash \phi \wedge \psi}\ \wedge\text{-R}.$$

It turns out that for the whole intuitionistic hereditary Harrop fragment, in which just $\wedge$, $\supset$ and $\forall$ may occur negatively (the *definite formulae*) and $\wedge$, $\vee$, $\supset$, $\forall$ and $\exists$ may occur positively (the *goal formulae*), the strategy of applying right rules wherever they are applicable is complete. Furthermore, it is possible, for such classes of definite formulae and goal formulae, to encode all necessary instances of left rules in such proof by a single left rule, the appropriate notion of resolution proof. This is achieved by rewriting the antecedents of the sequents in a certain *clausal form*. The reader is referred to [9] for the details of these points.

The main novelty in linear logic is that for the desired classes of definite formulae and goal formulae, it may be necessary, under certain circumstances, to apply a left rule in the middle of a sequence of right rules in order to maintain completeness. 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.,*

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

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 can 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.,*

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

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 can be classified.

Similar comments apply to the !-R rule, for which, due to the syntactic restrictions which are placed on the premiss of the rule, there are a larger number of exceptions. For example, consider the sequent $!\,\phi \otimes !\,\psi \vdash !\,\psi$, which has a proof containing an inference of the form

$$\cfrac{\cfrac{!\,\phi, !\,\psi \vdash \psi}{!\,\phi, !\,\psi \vdash !\,\psi}\ !\text{-R}}{!\,\phi \otimes !\,\psi \vdash !\,\psi}\ \otimes\text{-L}.$$

Again, we cannot apply the rules in the reverse order, as that would create a formula on the left without ! as its outermost connective.

One solution to this problem would be to exclude $\otimes$ and ! from the class of definite formulae in linear logic. However, this is both undesirable and unnecessary. It turns out that these exceptions to the strategy of applying right rules wherever they are applicable can be eliminated in the next step of our analysis, in which, just as in the setting of intuitionistic hereditary Harrop formulae, we rewrite the antecedents to a certain clausal form for which a single left rule, the appropriate notion of resolution rule, is complete.

Thus we show how an analysis of the permutability properties of the two-sided linear sequent calculus can be used to determine fragments of linear logic for which a suitable notion of resolution proof can be defined. A summary of this analysis consists in essentially two, mutually dependent, steps: (i) The identification of the class of formulae such that strategy of constructing proofs (considering rules to be reduction operators, in the sense of Kleene [7]) by applying right rules wherever they are applicable is complete, subject to certain exceptions to this strategy, which must be handled, *i.e.*, made goal-directed, by the second step. This class of proofs, constructed by the application of the right rules wherever possible, subject to the exceptions to be handled by step (ii), is the appropriate class of *uniform* proofs; (ii) The exploitation of the structural properties of the antecedents of linear sequents so that exceptions to the right rule-first strategy can be eliminated and so that we can invoke a single left rule, a *resolution* rule whilst retaining (soundness and) completeness with respect to linear sequent calculus. This is achieved by introducing a mapping which reduces antecedents to multisets of *clauses*, see Definitions 3.1 and 3.2.

Of course, there are other, more pragmatically motivated, influences on the choices of definite formulae, and indeed on the choices of goal formulae *q.v.* [10], [9]. We remark here that if we were to restrict definite formulae to be just the clauses permitted by (ii), above, we should be forced to constrain goal formulae rather more than is essential, see Definition 2.3. It remains an open problem to determine the maximal class of linear formulae for which suitable notions of uniform proof are complete.

In this paper we recapitulate parts of an earlier paper, [5], and extend the results to larger classes of formulae, thereby showing how larger fragments of linear logic can be used as logic programming languages.[2] In the next section we discuss the appropriate notion of uniform proof for linear logic. Section 3 discusses clauses, clausal decomposition and resolution. In Section 4 we discuss, rather briefly, some issues pertaining to the implementation of an interpreter for the language of Sections 3 and 4.

2 Uniform Linear Proofs

In the this section we develop an appropriate notion of uniform proof, together with compatible classes of definite formulae and goal formulae. We begin with the notion of *locally LR* proof: this captures those exceptions to the strategy of applying right rules wherever possible which can be handled by the reduction of antecedents to clausal form, *q.v.* Section 3.

DEFINITION 2.1 (LOCALLY LR PROOFS) *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*

[2] In particular, we extend the class of goals, G, to allow formulae of the form $!G$ and $?G$ and, more surprisingly, the class of programs to allow formulae of the form $D_1 ⅋ D_2$; also, both classes of formulae may contain certain occurrences of the linear constants. These formulae require substantial extensions of all of the main definitions of [5]; the corresponding results are correspondingly more complex. In particular, the definitions of uniform proof, clause, clausal decomposition and resolution proof are substantially more intricate.

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

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

$$\dfrac{\dfrac{\Xi}{\Gamma \vdash \Delta}\ !\text{-R}}{\Gamma' \vdash \Delta'}\ \sharp\text{-L}$$

where $\sharp$-L *is one of* $\otimes$-L, $\&$ -L, $\multimap$ -L, $\wedge$-L *and* $\maltese$ -L, *and* Ξ *is locally LR, or of the form*

$$\dfrac{\dfrac{}{\vdash \mathbf{1}}\ \mathbf{1}\text{-R}}{\Gamma \vdash \mathbf{1}}\ \sharp\text{-L}$$

where $\sharp$-L *is either* $\mathbf{1}$-L *or* $W!$-L, *or of the form*

$$\dfrac{\dfrac{}{\Gamma \vdash \top, \Delta}\ \top\text{-R}}{\Gamma' \vdash \Delta'}\ \sharp\text{-L}$$

where $\sharp$-L *is one of* $\mathbf{1}$-L, $\otimes$-L, $\&$ -L, $\multimap$ -L, $\wedge$-L, $!$-L, $W!$-L, $C!$-L *and* $\maltese$ -L. $\square$

However, as discussed in [5], 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 $\multimap$ -L rule. Hence we introduce the notion of a *simple* proof, *cf.* [12].

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

$$\dfrac{\begin{matrix}\Xi' \\ \Gamma \vdash \phi, \Delta\end{matrix} \qquad \psi \vdash \psi}{\Gamma, \phi \multimap \psi, \Gamma' \vdash \psi, \Delta, \Delta'} \qquad \square$$

The intuition behind simple proofs is that the right hand subproof used in the $\multimap$ -L rule should be trivial, *i.e.*, require no computational effort to prove. It is straightforward to show that all locally LR proofs can be written as proofs which are simple and locally LR proofs. Henceforth we shall refer to such simple locally LR proofs as *uniform proofs*.

Next we introduce the notions of *definite formulae* and *goal formulae*. Definite formulae and goal formulae are formulae that may occur in sequents respectively negatively and positively whilst retaining the completeness of uniform proofs. Note that these classes are strictly more general than the corresponding ones of [5].

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

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

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. $\square$

The reader should note that the class of implicational definite formulae may be extended to include formulae of the form $G \multimap L$, where L is of the form A, $\bigwedge x \,.\, L$, $L \,\&\, L$ (and others, see [6]). However, these extensions are of little interest in this paper and so we restrict our attention to implicational definite formulae of the form $G \multimap A$. Note also that as the necessary permutation properties for $\oplus$-L and ?-L fail, we exclude $\oplus$ and ? from the class of definite formulae: we also exclude $\bigvee$, the infinitary version of $\oplus$: its inclusion is difficult to justify computationally and is technically uninteresting. Note that formulae of the form $\exists x \,.\, D$ are not considered to be definite formulae in [9], [10].

The following lemma can be shown by a simple induction on the structure of proofs:

LEMMA 2.4 *Let Γ be a multiset of definite formulae and let Δ be a multiset of goal formulae. A proof of $\Gamma \vdash \Delta$ contains no occurrences of* $\bot$-L, $\oplus$-L, ?-L *or* $\bigvee$-L. $\Box$

Theorem 2.5, which can be shown by straightforward induction on the structure of proofs, summarizes the required properties of definite formulae and goal formulae.

THEOREM 2.5 *Let $\mathcal{P}$ be a multiset of definite formulae and let $\mathcal{G}$ be a multiset of goal formulae. Then $\mathcal{P} \vdash \mathcal{G}$ has a proof in linear sequent calculus iff $\mathcal{P} \vdash \mathcal{G}$ has a simple locally LR proof.* $\Box$

3 Resolution

In this section we develop the notion of resolution proof, Definition 3.10. Such proofs are goal-directed and require a single left rule, an appropriate form of resolution rule.

The definition of resolution proofs is well understood for a large fragment of linear logic, the details being given in earlier work of the present authors [5]. In particular, the classes of definite formulae and goal formulae considered there are given by

$$\begin{array}{lcl} D & ::= & A \mid D \,\&\, D \mid D \otimes D \mid G \multimap A \mid \bigwedge x \,.\, D \mid\, !D \\[1ex] G & ::= & A \mid A^{\bot} \mid G \otimes G \mid G \oplus G \mid G \,⅋\, G \mid G \,\&\, G \\ & & \mid D \multimap G \mid \bigwedge x \,.\, G \mid \bigvee x \,.\, G \,, \end{array}$$

where A ranges over atoms. Here we introduce additional formulae of the form $D \,⅋\, D$, $!\,G$ and $?\,G$. Each of these classes of formulae introduces substantial, related complications to the definition of resolution proof over that of [5]. Recall from Section 1 that we must rewrite the antecedent (a multiset of definite formulae) so that a single left rule will suffice; but recall also that such a rewriting must eliminate the need for the exceptions to the right rule-first strategy given in the definition of uniform proofs, *q.v.* Definition 2.1. In the setting of [5], as above, the definition of such a rewriting of antecedents is quite straightforward, with formulae of the form $!\,D$ providing the one difficult case. The decomposition $[-]$ of [5] rewrites programs, multisets of closed definite formulae, to a form with the property that the (multiplicative) conjunctions are eliminated by the multiset structure, the universally quantified variables are marked as global or local and the quantifiers omitted and the formulae under a ! are "flattened", so that the "equivalent" multiset of clauses consists of formulae given by

$$\begin{array}{lcl} B & ::= & A \mid G \multimap A \\ C & ::= & B \mid\, !\,(\underbrace{B \otimes \ldots \otimes B}_{n \text{ times}})\,. \end{array}$$ [3]

This means that clauses are bounded with respect to the depth of the outermost implication, in that such an implication occurs either outermost, as in $G \multimap A$, or as the outermost connective of a formula B_i in $!\,(B_1 \otimes \ldots \otimes B_n)$. This makes it simple to determine whether or not a given atom unifies with a given clause. In addition, it is a trivial matter to determine whether or not a given clause commences with !. In this setting, the resolution rule can be considered to obtain the following form:

$$\frac{\mathcal{D}[\vec{t}/\vec{x}]^g, [\vec{C}[\vec{t}/\vec{x}]], (!(G' \multimap A') \otimes \vec{C})[\vec{t}/\vec{x}]^g \vdash_R G\,, \mathcal{G} \qquad A \vdash_R A}{\mathcal{D}, !(G' \multimap A') \otimes \vec{C} \vdash_R A, \mathcal{G}}$$

where $G \multimap A = (G' \multimap A')[\vec{t}/\vec{x}]$, and where $\vec{C}$ denotes $C_1 \otimes \ldots \otimes C_m$ for $m \geq 0$. Note that we distinguish two forms of substitution, $[\vec{t}/\vec{x}]$ and $[\vec{t}/\vec{x}]^g$ (see [5]): this arises because of the structural properties of the exponential !: certain *global* variables in the program, those not guarded by a !, might be instantiated only once, whereas those *local* variables guarded by a ! might be duplicated by

[3] A related notion of decomposition is discussed in [11], p. 58.

contractions and so might receive different instantiations on different branches of the proof: the substitution form $[\vec{t}/\vec{x}]^g$ is used to distinguish the application of the substitution to just global variables.

The flattening clauses does not preserve linear equivalence if we admit positive occurrences of goal formulae of the form $!G$: for example, consider the formula $!(p \otimes !q)$. Under the definition of [5], we have $[\{!(p \otimes !q)\}] = \{!p, !q\}$, from which we can derive $!q$, whereas $!q$ is not derivable from $!(p \otimes !q)$ in linear sequent calculus: in the presence of goals of the form $!G$ we need a more subtle mapping.

Our solution is not to flatten the clauses, but leave them as they are, so that in the above example, we have $[\{!(p \otimes !q)\}] = \{!(p \otimes !q)\}$. This means that we may need to look arbitrarily deep inside a clause to determine if a given atom A unifies with that clause or not, which consequently makes the notion of resolution somewhat more complicated. Thus the earlier restriction that goal formulae may not be of the form $!G$ means that clauses have a particularly simple form and a correspondingly simple notion of resolution, whereas without this restriction, clauses and resolution become more involved, but, as we shall see, not unbearably so. Note that we maintain the property that we can decide immediately whether or not the !-R rule must be used.

The addition of $\parr$ to the class of definite formulae also requires an extension of the notion of resolution proof over that taken in [5]: this is because we must "encode" occurrences of the $\parr$-L rule into the definition of resolution proof. The latter notion is the more straightforward, in that we simply allow a clause to be of the form $C_1 \parr C_2$ where C_1 and C_2 are clauses. The notion of resolution can also be extended in a manner similar to that for non-flattened clauses. However, allowing clauses to be of the form $C_1 \parr C_2$ destroys the property that a clause either does not contain ! or has it as its outermost connective, a property that we noted was important for the ability to handle goals such as $!q$. For example, consider the sequent $!p \parr !q \vdash !p, !q$. This is provable, as we can derive the above sequent from $!p \vdash !p$ and $!q \vdash !q$ from a single instance of $\parr$-L. However, when considering a top-down search for a proof of this sequent, it is clear that the !-R rule is not applicable, as the sequent is not in the required syntactic form; so that in order to retain completeness we need a more sophisticated rule for goal formulae of the form $!G$ when $\parr$ may appear in programs.

To determine the precise nature of this rule, recall the notion of a simple locally LR proof, in which an occurrence of the !-R rule might be immediately followed by one of $\otimes$-L, $\&$-L, $\multimap$-L and $\parr$-L. Now such occurrences of the first three rules are accounted for in the mapping $[-]$, and hence need not be considered for clauses. An occurrence of !-R on the left of $\multimap$-L is precisely the resolution rule, and hence is not needed here, and !-R on the right of $\multimap$-L cannot occur when the antecedent is a clause, as the proof is simple. Hence, we need only concern ourselves with occurrences of $\parr$-L below !-R, in order to derive the appropriate rule for $!G$ in a resolution proof. The essential task is to determine whether a sequence of applications of $\parr$-L, when interpreted as a search operator, will allow the !-R rule to be applied. As we shall see, the key property of clauses in here will be that if a clause does not commence with a !, then it either does not contain ! or it commences with $\parr$.

Thus the addition of $!G$ to goals and $D_1 \parr D_2$ to programs causes us to use more sophisticated notions of a clause, resolution and the !-rule. It should be noted that the absence of one of these features is sufficient to allow significant simplifications of the procedure described below. In other words, to use $!G$ but not $D_1 \parr D_2$ means that the !-rule can be used directly. For example, the language of [6] does not allow the use of the connective $\parr$ at all. On the other hand to use $D_1 \parr D_2$ but not $!G$ means that only the notion of resolution need be extended, and, as we shall see, this in itself is not a vast complication.

Another possible approach would be to try to build the $\parr$-L rule into the mapping $[-]$, just as those for $\&$-L, $\otimes$-L and $\bigwedge$-L were. The main difficulty is that the $\parr$-L rule requires that the entire sequent by split, and hence we need to know the current goal before this can be done, and so it seems simpler to build the occurrences into the rules which need this action, rather than attempt to compile such a rule away. Another advantage of this "dynamic" approach to the encoding of the $\parr$-L rule is that the extra complexity is only introduced when it is necessary. As we shall see below, this is only when we are trying to prove a goal formula $!G$ from a program in which a $\parr$ occurs; when either of these features is absent, no extra complexity is needed.

We are now in a position to give the definition of a clause.

Definition 3.1 (Clause) *We define classes of* basic clauses, subclauses *and* clauses *by the following grammar:*

$$\begin{array}{llll}
\textit{Basic clauses} & B & ::= & \mathbf{1} \mid \bot \mid A \mid G \multimap A \\
\textit{Subclauses} & S & ::= & B \mid S \otimes S \mid S \parr S \mid\, !S \\
\textit{Clauses} & C & ::= & B \mid S \parr S \mid\, !S
\end{array} \qquad \Box$$

Definition 3.2 (Clausal Decomposition) *Let $\cup$ denote multiset union. We define a mapping $[-]$ from definite formulae to multisets of clauses as follows:*

$$\begin{array}{rcl@{\qquad}rcl}
[\mathcal{P}] & =_{def} & \bigcup_{D\in\mathcal{P}}[D] & [D_1 \,\&\, D_2] & =_{def} & [D_1] \text{ or } [D_2] \\
[\mathbf{1}] & =_{def} & \{\mathbf{1}\} & [D_1 \otimes D_2] & =_{def} & [D_1] \cup [D_2] \\
[\bot] & =_{def} & \{\bot\} & [\bigwedge x \,.\, D] & =_{def} & [D] \\
[A] & =_{def} & \{A\} & [!\,D] & =_{def} & \{!\bigotimes_{C\in[D]} C\} \\
[G \multimap A] & =_{def} & \{G \multimap A\} & & & \\
 & & & [D_1 \maltese D_2] & =_{def} & \{(\bigotimes_{C\in[D_1]} C) \maltese (\bigotimes_{C\in[D_2]} C)\}
\end{array}$$

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

Note that the definition of the decomposition $[-]$ for formulae of the form $D\&D$ is non-deterministic.[4] Just as described above [5], the distinction between local and global variables will facilitate the sound use of substitutions (which we suppose to be calculated by unification) in the definition of resolution proof, Definition 3.10.

Definition 3.3 (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.* □

Note that the marked "global" and "local" variables of Definition 3.2 satisfy this definition. For a given substitution $[\vec{t}/\vec{x}]$, we consider that denote that the application of the substitution to subclauses of the form $!S$ is to only those variables in S that are marked as global (*cf.* the notation $[\vec{t}/\vec{x}]$ and $[\vec{t}/\vec{x}]^g$ discussed above). Of course, due to the possibility of splitting the program during computation, this may mean updating variables simultaneously across several branches of the proof.[5] [6]

It should be clear that a proof of $[\mathcal{P}] \vdash \mathcal{G}$ can be easily transformed into a proof of $\mathcal{P} \vdash \mathcal{G}$ by the insertion of the appropriate left rules and the application of appropriate substitutions (to deal with the quantifiers).

Definition 3.4 (Antecedent Division) *Let $\cup$ denote multiset union, and let $\mathcal{D}$ be a multiset of definite formulae. We define two multisets $\mathcal{D}^E$ and $\mathcal{D}^L$ such that $\mathcal{D} = \mathcal{D}^E \cup \mathcal{D}^L$ as follows:*

$$\begin{array}{rcl}
\mathcal{D}^E & =_{\mathrm{def}} & \bigcup_{D\in\mathcal{D}}\{\, D \mid \textit{the outermost connective of } D \textit{ is } ! \textit{ or } D = \mathbf{1} \,\} \\
\mathcal{D}^L & =_{\mathrm{def}} & \bigcup_{D\in\mathcal{D}}\{\, D \mid \textit{the outermost connective of } D \textit{ is not } ! \,\}. \quad \square
\end{array}$$

Definition 3.5 (Succedent Division) *Let $\cup$ denote multiset union, and let $\mathcal{G}$ be a multiset of goal formulae. We define two multisets $\mathcal{G}^E$ and $\mathcal{G}^L$ such that $\mathcal{G} = \mathcal{G}^E \cup \mathcal{G}^L$ as follows:*

$$\begin{array}{rcl}
\mathcal{G}^E & =_{\mathrm{def}} & \bigcup_{G\in\mathcal{G}}\{\, G \mid \textit{the outermost connective of } G \textit{ is } ? \textit{ or } G = \bot \,\} \\
\mathcal{G}^L & =_{\mathrm{def}} & \bigcup_{G\in\mathcal{G}}\{\, G \mid \textit{the outermost connective of } G \textit{ is not } ? \textit{ and } G \textit{ is not } \bot \,\}. \quad \square
\end{array}$$

Definition 3.6 (Multiset Expansion) *Let $\mathcal{C}$ be a multiset and let $\cup$ denote multiset union. We define $\mathcal{C}^n$ for integers $n \geq 0$ inductively as follows: $\mathcal{C}^0 =_{\mathrm{def}} \emptyset$, $\quad \mathcal{C}^{n+1} =_{\mathrm{def}} \mathcal{C} \cup \mathcal{C}^n$.* □

Definition 3.7 (Expansion) *Let $\mathcal{D}$ be a multiset of clauses, let $\cup$ denote multiset union, and let $!C \in \mathcal{D}$. A $!C$-*expansion *of $\mathcal{D}$ is a multiset $\mathcal{D}'$ of clauses such that $\mathcal{D}' = \mathcal{D}^L$ or $\mathcal{D}' = \mathcal{D}^L \cup \{[C]^L\}^n \cup \mathcal{E}$ for some $n \geq 1$, where $\mathcal{E}$ is an expansion of $[C]^E$.* □

Resolution proofs are defined in Definition 3.10. As resolution proofs will be defined only for sequents in which the antecedent is a multiset of clauses and the succedent a multiset of goal formulae, we will use the notation $\vdash_R$ instead of $\vdash$ in sequents involving resolution proofs. It is convenient to use this notation in Definitions 3.8 and 3.9.[7]

---

[4] In terms of an interpreter (*q.v.* Section 4), this amounts to a certain *mutual exclusion* property.

[5] 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.

[6] The reader is referred to [5] for an extended discussion of of local and global variables.

[7] The following sequence of definitions is well-founded: our use of $\vdash_R$ here is merely a notational convenience.

Components, Definition 3.8, will be required in the notion of resolution proof when we come to a sequents of the form $\mathcal{D} \vdash_R G_1 \otimes G_2, \mathcal{G}$ and $\mathcal{D} \vdash_R !G, \mathcal{G}$. In the latter case, rather than merely checking whether $\mathcal{D}^L = \emptyset = \mathcal{G}^L$, (which amounts to a restatement of the syntactic condition on the premiss of the !-R rule), we look for a *component* of $\mathcal{D} \vdash_R \mathcal{G}$ for which this is true. Clearly, when there are no clauses of the form $C_1 \maltese C_2$ in the program, this procedure reduces to the !-R rule. Similar remarks apply to former case.

DEFINITION 3.8 (COMPONENT) *Let $\mathcal{D}$ be a multiset of clauses and $\mathcal{G}$ be a multiset of goal formulae. Let $C \in \mathcal{D}$. A* multiset of C-components *of the sequent $\mathcal{D} \vdash_R \mathcal{G}$ is a non-empty multiset of sequents satisfying at least one of the following*[8]*:*

1. *$\{ \mathcal{D} \vdash_R \mathcal{G} \}$ is a (singleton) multiset of C-components of the sequent $\mathcal{D} \vdash_R \mathcal{G}$ for any $C \in \mathcal{D}$ (we take the case in which $\mathcal{D} = \emptyset$ to be included here);*

2. *Let $!S \in \mathcal{D}$. $\{ \mathcal{D}, !S, \mathcal{D}' \vdash_R \mathcal{G} \}$ is a (singleton) multiset of $!S$-components of the sequent $\mathcal{D} \vdash_R \mathcal{G}$ where $!S$ is* not *of the form $!(S_1 \maltese S_2)$ if $\mathcal{D}'$ is a $!S$-expansion of $\mathcal{D}$;*

3. *Let C be a formula of the form $S_1 \maltese S_2$. Then $\mathcal{C}_1 \cup \mathcal{C}_2$ is a multiset of C-components of the sequent $\mathcal{D} \vdash_R \mathcal{G}$, where $C \in \mathcal{D}$, if $\mathcal{C}_1$ is a multiset of $[S_1]$-components of the sequent $\mathcal{D}^E, [S_1], \mathcal{D}_1 \vdash_R \mathcal{G}_1, \mathcal{G}^E$ and $\mathcal{C}_2$ is a multiset of $[S_2]$-components of the sequent $\mathcal{D}^E, [S_2], \mathcal{D}_2 \vdash_R \mathcal{G}_2, \mathcal{G}^E$, where $\mathcal{D}^L \backslash \{C\} = \mathcal{D}_1 \cup \mathcal{D}_2$ and $\mathcal{G}^L = \mathcal{G}_1 \cup \mathcal{G}_2$;*

4. *Let C be a formula of the form $!(S_1 \maltese S_2)$. Then $\mathcal{C}_1 \cup \mathcal{C}_2$ is a multiset of C-components of the sequent $\mathcal{D} \vdash_R \mathcal{G}$, where $C \in \mathcal{D}$, if $\mathcal{C}_1$ is a multiset of $[S_1]$-components of the sequent $\mathcal{D}^E, [S_1], \mathcal{D}_1 \vdash_R \mathcal{G}_1, \mathcal{G}^E$ and $\mathcal{C}_2$ is a multiset of $[S_2]$-components of the sequent $\mathcal{D}^E, [S_2], \mathcal{D}_2 \vdash_R \mathcal{G}_2, \mathcal{G}^E$, where $\mathcal{D}^L = \mathcal{D}_1 \cup \mathcal{D}_2$ and $\mathcal{G}^L = \mathcal{G}_1 \cup \mathcal{G}_2$.*

Where it is notationally convenient, we will write just component *for C-component.* □

For example, $\{p \vdash_R p\} \cup \{q \vdash_R q\}$ is a multiset of components of the sequent $p \maltese q \vdash_R p, q$. Similarly, $\{!(p \maltese q), p \vdash_R p\} \cup \{!(p \maltese q), p \vdash_R p\} \cup \{!(p \maltese q), q, q \vdash_R q \otimes q\}$ is a multiset of components of the sequent $!(p \maltese q) \vdash_R p, p, q \otimes q$. (See also example (2) (below).)

Note that components are the appropriate generalization of the notion of *expansion* in [5]. Note also that the components of $\mathcal{P}, !D \vdash_R \mathcal{G}$ and of $\mathcal{P}, !D, !D \vdash_R \mathcal{G}$ coincide, and that any component of $\mathcal{P} \vdash_R \mathcal{G}$ is also a component of $\mathcal{P}, !D \vdash_R \mathcal{G}$. We remark that our ability to concentrate all relevant occurrences of the $\maltese$-L rule into the notion of component relies on the permutability of $\maltese$-L with respect to certain other left rules.

Below we introduce the notion of a resolvant. This is needed in the definition of the resolution rule because a clause may be of an arbitrary depth, it thereby not being obvious from the outermost structure of the clause which atoms will match it.

DEFINITION 3.9 (RESOLVANT) *Let $\mathcal{D}$ range over multisets of definite formulae and $\mathcal{G}$ range over multisets of goal formulae. A* resolvant, *$\mathcal{R}$, of the sequent $\mathcal{D} \vdash_R G, \mathcal{G}$ is a non-empty multiset of sequents satisfying at least one of the following:*

1. *$\mathcal{R}$ is a multiset $\{\mathcal{D}_0 \vdash_R A, \mathcal{G}_0,\ \mathcal{D}_1 \vdash_R \mathcal{G}_1, \ldots, \mathcal{D}_n \vdash_R \mathcal{G}_n\}$ of components of $\mathcal{D} \vdash_R A, \mathcal{G}$ such that $\mathcal{D}_0 \vdash_R A, \mathcal{G}_0$ is initial;*

2. *$\mathcal{R}$ is the multiset $\{\mathcal{D}_0[\vec{t}/\vec{x}] \vdash_R G'[\vec{t}/\vec{x}], \mathcal{G}_0,\ \mathcal{D}_1[\vec{t}/\vec{x}] \vdash_R \mathcal{G}_1, \ldots, \mathcal{D}_n[\vec{t}/\vec{x}] \vdash_R \mathcal{G}_n\}$, where $\{\mathcal{D}_0, G' \multimap A' \vdash_R A, \mathcal{G}_0,\ \mathcal{D}_1 \vdash_R \mathcal{G}_1, \ldots, \mathcal{D}_n \vdash_R \mathcal{G}_n\}$ is a multiset of components of $\mathcal{D} \vdash_R A, \mathcal{G}$ and $A = A'[\vec{t}/\vec{x}]$.* □

We now come to the definition of a resolution proof. Where no confusion can arise, we often write just "$\mathcal{D} \vdash_R \mathcal{G}$" rather than "$\mathcal{D} \vdash_R \mathcal{G}$ is provable" or "$\mathcal{D} \vdash_R \mathcal{G}$ has a resolution proof". Resolution proofs are goal-directed and employ a single left rule, the *resolution* rule (*2*) (below). Resolution proofs provide mechanisms neither for calculating appropriate splittings of antecedents and succedents nor for calculating expansions. Such calculations should be handled at the level of the definition of an interpreter, *q.v.* Section 4.

[8] The first two clauses of this definition are required by those instances of the $\otimes$- and !-rules of resolution proofs, Definition 3.10, which do not involve $\maltese$.

Definition 3.10 (Resolution Proof) *Let $\mathcal{D}$ range over multisets of definite formulae and let $\cup$ denote multiset union. A* **resolution proof** *is tree regulated by the following rules:*

1. *The axiom judgement is given by:*
$$\overline{\mathcal{D} \vdash_R \mathcal{G}},$$
where one of the following conditions holds:

 i. $\mathcal{D}^L = \{A'\}$, $\mathcal{G}^L = \{A\}$ *and* $A = A'[\vec{t}/\vec{x}]$;
 ii. $\mathcal{D}^L = \emptyset$, $\mathcal{G}^L = \{A\}$ *and there exists* $!A' \in \mathcal{D}^E$ *such that* $A = A'[\vec{t}/\vec{x}]$;
 iii. $\mathcal{D}^L = \emptyset$ *and* $\mathcal{G}^L = \{\mathbf{1}\}$;
 iv. $\mathcal{D}^L = \{\bot\}$ *and* $\mathcal{G}^L = \emptyset$;
 v. $\top \in \mathcal{G}$;

2. *We shall refer to this rule as the* resolution *rule:*
$$\frac{\mathcal{D}_1 \vdash_R \mathcal{G}_1 \;\ldots\; \mathcal{D}_n \vdash_R \mathcal{G}_n}{\mathcal{D} \vdash_R A, \mathcal{G}},$$
where $\bigcup_{i=1}^n \{\mathcal{D}_i \vdash_R \mathcal{G}_i\}$ is a resolvant of $\mathcal{D} \vdash_R A, \mathcal{G}$;

3. *The $R^\perp$-rule:*
$$\frac{\mathcal{D} \vdash_R \mathcal{G}}{\mathcal{D} \vdash_R \bot, \mathcal{G}}$$

4. *The $\bot$-rule:*
$$\frac{\mathcal{D}, [D] \vdash_R \mathcal{G}}{\mathcal{D} \vdash_R D^\perp, \mathcal{G}}$$

5. *The $\otimes$-rule:*
$$\frac{\mathcal{D}_0 \vdash_R G_1, \mathcal{G}_0 \quad \mathcal{D}_0' \vdash_R G_2, \mathcal{G}_0' \quad \mathcal{D}_1 \vdash_R \mathcal{G}_1 \;\ldots\; \mathcal{D}_n \vdash_R \mathcal{G}_n}{\mathcal{D} \vdash_R G_1 \otimes G_2, \mathcal{G}},$$
where $C \in \mathcal{D}$ and $\{\mathcal{D}_0, \mathcal{D}_0' \vdash_R G_1 \otimes G_2, \mathcal{G}_0, \mathcal{G}_0'\} \cup \bigcup_{i=1}^n \{\mathcal{D}_i \vdash_R \mathcal{G}_i\}$ is a multiset of C-components of the sequent $\mathcal{D} \vdash_R G_1 \otimes G_2, \mathcal{G}$;

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:*
$$\frac{\mathcal{D}, [D] \vdash_R G, \mathcal{G}}{\mathcal{D} \vdash_R D \multimap G, \mathcal{G}}$$

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

12. *The ?-rule:*
$$\frac{\mathcal{D} \vdash_R \{G\}^n, ?\,G, \mathcal{G}}{\mathcal{D} \vdash_R ?\,G, \mathcal{G}},$$
for some $n \geq 1$;

13. *The !-rule:*
$$\frac{\mathcal{D}_0 \vdash_R G, \mathcal{G}_0 \quad \mathcal{D}_1 \vdash_R \mathcal{G}_1 \;\ldots\; \mathcal{D}_n \vdash_R \mathcal{G}_n}{\mathcal{D} \vdash_R !G, \mathcal{G}},$$
where $C \in \mathcal{D}$ and $\{\mathcal{D}_0 \vdash_R G, \mathcal{G}_0\} \cup \bigcup_{i=1}^n \{\mathcal{D}_i \vdash_R \mathcal{G}_i\}$ is a multiset of C-components of $\mathcal{D} \vdash_R !G, \mathcal{G}$ and $\mathcal{D}_0^L = \emptyset = \mathcal{G}_0^L$.

Note that Rules 1 and 2 affect other sequents in the proof, so that a resolution proof is well-defined only if all occurrences of Rules 1 and 2 yield compatible substitutions. □

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

$$\frac{c, p, p \multimap r \vdash_R r \qquad c, q_1, q_2, (q_1 \otimes q_2) \multimap s \vdash_R q_1 \otimes q_2}{p, q_1, q_2, !((p \multimap r) ⅋ ((q_1 \otimes q_2) \multimap s)) \vdash_R r, s}, \tag{1}$$

and that the following is an example of the $\otimes$-rule (*5*) (above):

$$\frac{!(p \maltese q), p \vdash_R p \quad !(p \maltese q), p \vdash_R p \quad !(p \maltese q), q \vdash_R q \quad !(p \maltese q), q \vdash_R q}{!(p \maltese q) \vdash_R p, p, q \otimes q}. \tag{2}$$

The following two lemmas can be shown by induction on the structure of resolution proofs:

LEMMA 3.11 (WEAKENING AND CONTRACTION) *Let $\mathcal{P}$ be a multiset of mixed clauses, $\mathcal{G}$ be a multiset of goal formulae, C be a clause and G be a goal formula. Then:*

1. *If $\mathcal{P} \vdash_R \mathcal{G}$ then $\mathcal{P}, !C \vdash_R \mathcal{G}$;*
2. *If $\mathcal{P} \vdash_R \mathcal{G}$ then $\mathcal{P} \vdash_R ?G, \mathcal{G}$;*
3. *If $\mathcal{P}, !C, !C \vdash_R \mathcal{G}$ then $\mathcal{P}, !C \vdash_R \mathcal{G}$;*
4. *If $\mathcal{P} \vdash_R ?G, ?G, \mathcal{G}$ then $\mathcal{P} \vdash_R ?G, \mathcal{G}$;*
5. *If $\mathcal{P}, \{C\}^n \vdash_R \mathcal{G}$ then $\mathcal{P}, !C \vdash_R \mathcal{G}$;*
6. *If $\mathcal{P} \vdash_R \{G\}^n, \mathcal{G}$ then $\mathcal{P} \vdash_R ?G, \mathcal{G}$.* □

LEMMA 3.12 *Let $\mathcal{P}$ be a multiset of definite formulae and let $\mathcal{G}$ be a multiset of goal formulae. Then $[\mathcal{P}] \vdash_R \mathcal{G}$ has a resolution proof iff there is multiset of components $\mathcal{C}$ of $[\mathcal{P}] \vdash_R \mathcal{G}$ such that each element of $\mathcal{C}$ has a resolution proof.*

PROOF SKETCH The forward direction is obvious. For the other direction, we proceed by induction on the height of the shortest resolution proof of the individual components. Let $\mathcal{C}$ be $\{\mathcal{P}_1 \vdash_R \mathcal{G}_1\} \cup \ldots \cup \{\mathcal{P}_n \vdash_R \mathcal{G}_n\}$ and for simplicity assume that the sequent with the shortest resolution proof is $\mathcal{P}_1 \vdash_R \mathcal{G}_1$. We illustrate the argument by giving an interesting case. Suppose the last rule applied is the $\otimes$-rule. Then we have that

$$\mathcal{P}_0 \vdash_R G_1, \mathcal{G}_0 \quad \mathcal{P}'_0 \vdash_R G_2, \mathcal{G}'_0 \quad \mathcal{P}'_1 \vdash_R \mathcal{G}'_1 \ldots \mathcal{P}'_m \vdash_R \mathcal{G}'_m$$

have resolution proofs, where $\{\mathcal{P}_0, \mathcal{P}'_0 \vdash_R G_1 \otimes G_2, \mathcal{G}_0, \mathcal{G}'_0\} \cup \{\mathcal{P}'_1 \vdash_R \mathcal{G}'{}_1\} \cup \ldots \cup \{\mathcal{P}'_m \vdash_R \mathcal{G}'_m\}$ is a multiset of components of $\mathcal{P}_1 \vdash_R \mathcal{G}_1$. Now, $\{\mathcal{P}_0, \mathcal{P}'_0 \vdash_R G_1 \otimes G_2, \mathcal{G}_0, \mathcal{G}'_0\} \cup \{\mathcal{P}'_1 \vdash_R \mathcal{G}'{}_1\} \cup \ldots \cup \{\mathcal{P}'_m \vdash_R \mathcal{G}'_m\} \cup \bigcup_{i=2}^{n} \{\mathcal{P}_i \vdash_R \mathcal{G}_i\}$ is a multiset of components of $\mathcal{P} \vdash_R G_1 \otimes G_2, \mathcal{G}'$, so by the hypothesis we have that $\mathcal{P} \vdash_R G_1 \otimes G_2, \mathcal{G}'$ has a resolution proof. □

We are now in a position to present the main theorem.

THEOREM 3.13 *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 in linear sequent calculus iff $[\mathcal{P}] \vdash_R \mathcal{G}$ has a resolution proof.*

PROOF SKETCH Suppose that $[\mathcal{P}] \vdash_R \mathcal{G}$ is provable. We show that $\mathcal{P} \vdash \mathcal{G}$ is provable in linear sequent calculus by induction on the structure of resolution proofs. We illustrate the argument by giving an interesting case. Suppose the last rule applied is the $\otimes$-rule. Then we have that

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

where $\{\mathcal{P}_0, \mathcal{P}'_0 \vdash_R G_1 \otimes G_2, \mathcal{G}_0, \mathcal{G}'_0\} \cup \bigcup_{i=1}^{n} \{\mathcal{P}_i \vdash_R \mathcal{G}_i\}$ is a multiset of components of $[\mathcal{P}] \vdash_R G_1 \otimes G_2, \mathcal{G}$, and so by the hypothesis we have that $\mathcal{D}_0 \vdash G_1, G_0$, $\mathcal{D}'_0 \vdash G_2, \mathcal{G}'_0$ and each $\mathcal{D}_i \vdash \mathcal{G}_i$ has a linear proof, where $[\mathcal{D}_0] = \mathcal{P}_0$, $[\mathcal{D}'_0] = \mathcal{P}'_0$, and $[\mathcal{D}_i] = \mathcal{P}_i$. Clearly then $\mathcal{D}_0, \mathcal{D}'_0 \vdash G_1 \otimes G_2, \mathcal{G}_0, \mathcal{G}'_0$ is provable, and we may then apply $\maltese$-L to reduce the number of components, and then insert a sequence of left rules, possibly including $\maltese$-L, into the proof to get a linear proof of $\mathcal{P} \vdash G_1 \otimes G_2, \mathcal{G}$.

Conversely, suppose that $\mathcal{P} \vdash \mathcal{G}$ is provable in linear sequent calculus. We show that $[\mathcal{P}] \vdash_R \mathcal{G}$ is provable by induction on the structure of proofs in linear sequent calculus. By Theorem 2.5, we may assume that the proof of $\mathcal{P} \vdash \mathcal{G}$ is locally LR. We illustrate the argument by giving an interesting case. Suppose the last rule applied is $\maltese$-L. Then the previous sequents are of the form

$$\mathcal{D}_1, D_1 \vdash \mathcal{G}_1 \quad \mathcal{D}_2, D_2 \vdash \mathcal{G}_2,$$

where $\mathcal{D}_1 \cup \mathcal{D}_2 \cup \{D_1 \maltese D_2\} = \mathcal{P}$ and $\mathcal{G}_1 \cup \mathcal{G}_2 = \mathcal{G}$, and by the hypothesis we have that $[\mathcal{D}_1], [D_1] \vdash_R \mathcal{G}_1$ and $[\mathcal{D}_2], [D_2] \vdash_R \mathcal{G}_2$. Hence, by Lemma 3.12, we have that $[\mathcal{D}_1], [\mathcal{D}_2], [D_1 \maltese D_2] \vdash_R \mathcal{G}_1, \mathcal{G}_2$, *i.e.*, $[\mathcal{P}] \vdash_R \mathcal{G}$. □

4 A Sketch of an Interpreter

In this section we sketch the design of an interpreter for linear logic programs. Resolution proofs provide the logical basis for proof-search in that they have the following properties: (i) Goal directed: we reduce on the right wherever possible; (ii) Employ a single left rule: resolution; (iii) Build-in the structural rules of weakening and contraction. However, the definition of resolution proof provides mechanisms neither for calculating appropriate splittings of antecedents and succedents nor for calculating expansions. For example, in the $\otimes$- and resolution rules it is necessary to calculate suitable sets of components and resolvants, respectively.

Our solution is to adopt a *lazy* approach, and to this end we permit an interpreter to construct *proto-proofs* by modifying the rules of resolution proof to delay the calculation of splittings (so that, for example, all suitable formulae in the antecedent and succedent go each way at a $\otimes$-rule). In order to maintain the soundness, we introduce the notion of *path* in such proto-proofs. We sketch the definition of the construction of a *path*[9] in such a proto-proof as follows: (I) The endsequent is in every path; (II) Traverse the proto-proof tree towards the leaves, starting at the endsequent: (i) Whenever a &-rule is reached, choose a branch and proceed; (ii) Whenever a $\otimes$- or resolution rule is reached, proceed along all branches; (III) Continue until all branches of the path have reached a leaf. The proto-proof determines a proof just in case for each possible path in it, there are expansions (at the appropriate rule applications) of the antecedent and succedent that are compatible with leaves in the path.

We illustrate this technique by considering two examples, (3) and (4) (below). Suppose we are faced with the endsequent $!(p \otimes q) \vdash (q \multimap (q \otimes (p \otimes q)))\ \&\ (p \otimes q)$, which is provable in linear sequent calculus. We must construct a resolution proof of $[!(p\otimes q)](=!(p\otimes q)) \vdash_R (q \multimap (q \otimes (p \otimes q)))\ \&\ (p \otimes q)$. Adopting our lazy approach, we obtain the proto-proof given below:

$$\dfrac{\dfrac{\dfrac{!(p\otimes q), q \vdash^{\sharp}_R q \qquad \dfrac{!(p\otimes q), q \vdash^{\sharp}_R p \qquad !(p\otimes q), q \vdash^{\sharp}_R q}{!(p\otimes q), q \vdash^{\sharp}_R p \otimes q}\,\otimes}{!(p\otimes q), q \vdash^{\sharp}_R q \otimes (p \otimes q)}\,\otimes}{!(p\otimes q) \vdash^{\sharp}_R q \multimap (q \otimes (p \otimes q))}\,\multimap \qquad \dfrac{!(p\otimes q) \vdash^{\flat}_R p \qquad !(p\otimes q) \vdash^{\flat}_R q}{!(p\otimes q) \vdash^{\flat}_R p \otimes q}\,\otimes}{!(p\otimes q) \vdash^{\sharp\flat}_R (q \multimap (q \otimes (p \otimes q)))\ \&\ (p \otimes q)}\,\& \tag{3}$$

In (2), there are two possible paths, one marked by $\sharp$ and one marked by $\flat$. The path marked by $\sharp$ is the more complicated. When calculating suitable splittings and expansions for this path, we must first use up the purely linear part of the antecedent, namely q, before considering the existence of a suitable expansion of $!(p \otimes q)$. Suppose that this q is used in the leftmost leaf, then it remains for us to check the existence of an expansion that is compatible with the remaining leaves, with q deleted from the antecedent, *i.e.,* $!(p \otimes q) \vdash^{\sharp}_R p$ and $!(p \otimes q) \vdash^{\sharp}_R q$. Here we see that the expansion $\{p, q\}$ is compatible with these leaves, so that we can conclude that this proto-proof does indeed determine a resolution proof of the given endsequent.

Our second example (4) of the use of paths is based on (1):

$$\dfrac{\dfrac{c, p, q_1, q_2, p \multimap r \vdash^{\sharp}_R p}{c, p, q_1, q_2, p \multimap r \vdash^{\sharp}_R r}\,\text{res.} \qquad \dfrac{c, p, q_1, q_2, (q_1 \otimes q_2) \multimap s \vdash^{\sharp}_R q_1 \qquad c, p, q_1, q_2 \vdash^{\sharp}_R q_2}{c, p, q_1, q_2, (q_1 \otimes q_2) \multimap s \vdash^{\sharp}_R q_1 \otimes q_2}\,\otimes}{p, q_1, q_2, !((p \multimap r) \maltese ((q_1 \otimes q_2) \multimap s)) \vdash^{\sharp}_R r, s}\,\text{res.} \tag{4}$$

There is just one path (marked by $\sharp$) in this proto-proof and the reader might easily verify that suitable splittings and expansions exist for a resolution proof to be determined.

We remark that it is important in the design of an interpreter to consider the permutability properties of right rules with respect to each other. The order of reduction of the formulae in the succedent should respect the relative permutability of the right rules. A complete description of an interpreter will be given in forthcoming work by the present authors. We remark that the "input/output model of resource consumption", introduced in [6], can be extended to implement our notion of path. The theory of paths is closely related to that of *proof nets* [3].

References

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

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

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

[4] Girard, J.-Y., Lafont, Y., Taylor, P. *Proofs and Types.* Cambridge University Press, 1989.

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

[5] Harland, James and Pym, David. *The Uniform Proof-theoretic Foundation of Linear Logic Programming (Extended Abstract).* Proceedings of the International Logic Programming Symposium, San Diego, October 1991, pp. 304-318. MIT Press, 1991. Preliminary version available as Report ECS-LFCS-90-124, University of Edinburgh, November 1990.

[6] 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, 1991.

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

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

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

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

[11] Read, S. *Relevant Logic.* Basil Blackwell, 1988.

A Linear Sequent Calculus

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.

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

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

$$\frac{\Gamma \vdash \Delta}{\Gamma, \mathbf{1} \vdash \Delta}\ \mathbf{1}\text{-L} \qquad\qquad \frac{}{\vdash \mathbf{1}}\ \mathbf{1}\text{-R}$$

$$\frac{}{\bot \vdash}\ \text{L}^{\bot} \qquad\qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}\ \text{R}^{\bot}$$

$$\frac{}{\Gamma, \mathbf{0} \vdash \Delta}\ \mathbf{0}\text{-L} \qquad\qquad \frac{}{\Gamma \vdash \top, \Delta}\ \top\text{-R}$$

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

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

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

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

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

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

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

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

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

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

$$\frac{\Gamma, \phi[t/x] \vdash \Delta}{\Gamma, \bigwedge x \,.\, \phi \vdash \Delta}\ \textstyle\bigwedge\text{-L} \qquad\qquad \frac{\Gamma \vdash \phi[y/x], \Delta}{\Gamma \vdash \bigwedge x \,.\, \phi, \Delta}\ \textstyle\bigwedge\text{-R}\ (y \text{ not free in } \Gamma,\ \Delta)$$

$$\frac{\Gamma, \phi[y/x] \vdash \Delta}{\Gamma, \bigvee x \,.\, \phi \vdash \Delta}\ \textstyle\bigvee\text{-L}\ (y \text{ not free in } \Gamma,\ \Delta) \qquad\qquad \frac{\Gamma \vdash \phi[t/x], \Delta}{\Gamma \vdash \bigvee x \,.\, \phi, \Delta}\ \textstyle\bigvee\text{-R}$$