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

# Logic Programming via Proof-Valued Computations

by

David J Pym and Lincoln A Wallen

LFCS Report Series
ECS-LFCS-92-246
November, 1992

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

In: ALPUK92: Proceedings of 4th UK Conference on Logic Programming, London, 1992 (K. Broda, editor). pp. 253-262. WICS, Springer, 1992. Please refer to this publication only.

# LOGIC PROGRAMMING VIA PROOF-VALUED COMPUTATIONS*

David J. Pym
University of Edinburgh
Scotland, U.K.

Lincoln A. Wallen
University of Oxford
England, U.K.

**Abstract**

We argue that the computation of a logic program can be usefully divided into two distinct phases: the first being a proof-valued computation or proof-search; the second a residual computation, or answer extraction. Extension of extraction techniques to various theories then permits more extensive languages and proof procedures to be employed for the computational solution of problems.

We illustrate these ideas with a simple propositional logic and show that SLD-resolution computes presentations of proofs in which the residual computation may be interleaved with the proof-search, whereas a more general proof procedure yields shorter presentations of (the same) proofs, but which require more extensive residual computations.

## 1 Introduction

One often takes the result of a computation of a logic program $(\Gamma; \exists x\psi)$ to be a substitution of a term $t$ for the (existentially quantified) variable of the query such that

$$\Gamma \quad \text{entails} \quad \psi[t/x];$$

indeed logic programs are usually modelled as sets of such substitutions [8]. This is a natural view to take since the substitutions typically carry information required by the user of the program.

However, since logical consequence is an abstract notion, our access to it is via concrete notions of deduction; *i.e.,* we construct a (finitary) proof that $\exists x\psi$ follows from $\Gamma$, and extract a term $t$ from this proof. There are thus two distinct phases of computation:

- *proof-search:* the computation of a proof; and

*This work was supported in part by ESPRIT BRA, "Logical Frameworks"; and U.K. SERC grant GR/G 58588, "Logical and Semantical Frameworks".

- *residual computation:* the extraction of witnesses from that proof.

It is obvious that the residual computation is dependent on the result of the proof-search.

The output from the first computation (*i.e.*, the proof) is, we believe, more naturally seen as the result of the computation; so from this point of view *any* sound proof procedure for a logical language gives rise to a notion of computation, the values computed being proofs.[1] The belief that this view is inadequate as a basis for (the theory of) logic programming stems, it seems, from the belief that it is easier to program with predicates and objects satisfying them, than with formulae and proofs proving them. If one considers logical systems as represented in the LF [4], the calculation of objects satisfying predicates and of proofs proving formulae is achieved by the same mechanism.[2]

If we require that the residual computation be an operation of low complexity, severe restrictions on both the language for expressing our programs and on the proof procedures used as interpreters for them must be accepted. Horn clauses and SLD-resolution are perhaps the best known matching pair (though see the work of Miller *et al.* [11, 12]). However, extraction techniques have been developed for various theories (see for example [1, 18]) and it is the extent of this knowledge that sets a limit on the languages and proof procedures that can be used in the first phase of computation.

In this paper we begin an investigation of logic programming (or, more abstractly, proof-search) as proof-valued computation. We show that even in a simple propositional language well-known methods of proof-search compute interesting values. More specifically, that a cut-free sequent calculus is naturally interpreted as computing *constructions* of natural deduction proofs. These constructions are non-normal in the sense that they present normal natural deductions in a non-normal, and often more compact, manner. Such constructions require further *residual* computation to bring them to normal form. Although we do not treat quantifiers here, it becomes clear that witnesses for existential quantifiers would, in general, fail to be explicit in such constructions. The residual computation is effectively the second phase of answer extraction referred to above.

We show that an analytic (or cut-free) sequential formulation of SLD-resolution computes constructions requiring limited residual computation; so limited, in fact, that the residual computation may be safely interleaved with the proof-search. In a quantified setting this would yield, as expected, explicit witnesses on termination.

In further work, we hope to apply the techniques of this paper to logics determined by general, schematic classes of natural deduction rules [10, 15, 16].

[1] An important, yet informal, constraint on such proof procedures is that their behaviour be predictable enough for programmers to appreciate the relationship between the form of their axiomatisations and the behaviour of the proof procedure in its attempt find a solution.

[2] An elementary notion of logic programming for the type theory of the LF is discussed in [16, 17]; a more elaborate notion is developed in [13].

Apart from its theoretical interest this approach to "computation via deduction" has a number of practical applications. One is to speed up computations that rely on Boolean (*i.e.,* ground) subcomputations. There is no need to construct normal proofs of such goals since answer extraction is not required. A second application is to support the integration of answer extraction and program extraction techniques. Extending a language with functional forms (say recursion or combinators for conditionals) permits the expression of answers to more complex queries and databases (*cf.* Gödel's Dialectica interpretation for Peano Arithmetic [3]). Such representations of answers may be manipulated, stored or even passed as input to other computations. In effect we are gaining a form of *modularity* in the expression (and computation) of answers. These ideas will be explored elsewhere.

## 2 Logical Background

It is well-known that, for certain subsystems of predicate logic, proofs of a sequent $\Gamma;\varphi$ within a (single-conclusioned) sequent calculus ($\boldsymbol{L}$) can be interpreted as constructing natural deductions ($\boldsymbol{N}$) of the succedent formula $\varphi$ from the antecedent formulae $\Gamma$ [19]. Let us denote this interpretation of sequent derivations by $[\![.]\!]^{E}$. Let $\boldsymbol{L}(l)$ denote the set of sequent derivations ($\vdash$) of $l$, and $\boldsymbol{N}(\Gamma;\varphi)$ the set of natural deductions ($\models$) of $\varphi$ from $\Gamma$; *i.e.,*

$$\begin{aligned} \boldsymbol{L}(\Gamma;\varphi) &:= \left\{d \mid \vdash^{d} \Gamma;\varphi\right\} \\ \boldsymbol{N}(\Gamma;\varphi) &:= \left\{M \mid \Gamma \models^{M} \varphi\right\}. \end{aligned}$$

Then, for all sequents $l$,

$$d \in \boldsymbol{L}(l) \quad \Rightarrow \quad [\![d]\!]^{E} \in \boldsymbol{N}(l). \tag{1}$$

The interpretation induces an equivalence relation $\sim$ on $\boldsymbol{L}(l)$ by

$$d \sim d' \qquad \text{iff} \qquad [\![d]\!]^{E} = [\![d']\!]^{E}. \tag{2}$$

If we view sequent derivations as natural deduction-valued functions it is natural to call this an *extensional* equivalence relation.

The fact that distinct sequent derivations are extensionally equivalent in this sense has been used as a criticism of the sequent calculus and of predicate logic itself [2].

After defining this interpretation for a pure implicational language below, in § 5 we introduce an alternative interpretation, $[\![.]\!]^{S}$, which emphasises some intensional properties of derivations as *construction operations.* As an application of this (constructive) interpretation we show that a standard cut-free sequent calculus for the logic constructs *normal* natural deductions in a *non-normal* (and hence shorter) way. The presentation below follows [4].

A natural deduction system $N$, for a *pure implicational propositional logic* is defined as follows. Let $n > 0$ be a fixed natural number. The set $\mathcal{F}$ of *formulae* is the smallest set containing the *propositional variables* $p_0, \ldots, p_n$ and such that $\varphi \supset \psi \in \mathcal{F}$ whenever $\varphi, \psi \in \mathcal{F}$. We shall use $\varphi, \psi$ and $\chi$ (possibly subscripted) to denote formulae.

Given a countably infinite set of *assumption markers*, $\{\xi_i\}_{i<\omega}$, the set of *proof expressions* $\mathcal{N}$ is the smallest set containing $\text{HYP}_\varphi(\xi)$, $\text{IMP-I}_{\varphi,\psi}(\xi : M)$ and $\text{IMP-E}_{\varphi,\psi}(M, N)$ whenever $\varphi, \psi \in \mathcal{F}$, $\xi$ is a marker and $M, N \in \mathcal{N}$. All occurrences of $\text{HYP}_\varphi(\xi)$ in the expression $M$ in $\text{IMP-I}_{\varphi,\psi}(\xi : M)$ are considered bound; such bound occurrences of marked hypotheses correspond to discharged assumptions. Capture-avoiding substitution of expressions for free occurrences of subexpressions of the form $\text{HYP}_\varphi(\xi)$ is defined as usual. The *end formula* of a proof expression of the first kind is $\varphi$; of the second kind is $\varphi \supset \psi$; and of the third kind is $\psi$. If we want to emphasise the end formula $\varphi$ of a proof expression $M$ we shall write it as a superscript thus $M^\varphi$.

A *(proof) context* is a finite sequence of *declarations* of the form $\xi : \varphi$. Equality on contexts is taken up to permutation. We shall use $\Delta$, possibly primed, to denote contexts. The *domain* of a context $\Delta = (\xi_i : \varphi_i)_{i<n}$, written $\text{dom}\Delta$, is the set of markers $\{\xi_i \mid i < n\}$. A context is said to be *well-formed* if its markers are pairwise distinct.

The set of well-formed context-expression pairs $(\Delta, M)$ is the smallest set satisfying:

N1. $(\Delta, \text{HYP}_\varphi(\xi))$ is well-formed whenever $\Delta$ is well-formed and $\xi : \varphi \in \Delta$;

N2. $(\Delta, \text{IMP-I}_{\varphi,\psi}(\xi : M)^{\varphi \supset \psi})$ is well-formed whenever $(\Delta, \xi : \varphi, M^\psi)$ is well-formed;

N3. $(\Delta, \text{IMP-E}_{\varphi,\psi}(M, N)^\psi)$ is well-formed whenever $(\Delta, M^{\varphi \supset \psi})$ and $(\Delta, N^\varphi)$ are well-formed.

We write $\Delta \models^M \varphi$ if $(\Delta, M^\varphi)$ is well-formed. It is clear that in such a case $M$ is a derivation of its end formula from the formulae of $\Delta$ in the usual sense; derivations being natural deductions.

Note that our presentation of propositions and their natural deduction proofs differs from that of the Curry-Howard correspondence. We remark only that the generality of our analysis requires the distinction made by our definitions between markers in contexts and the variables of lambda terms.

## 3 Sequents, calculi and logic programming

A *sequent* is a (well-formed) context-formula pair, written $\Gamma; \varphi$. In terms of our sequential understanding of logic programming, the context or antecedent of the sequent, $\Gamma$, corresponds to the *program* and the succedent, $\varphi$, to the *goal*. Typically, $\Gamma$ is a set of *clauses* and $\varphi$ is of the form $\exists x \psi$. Such a sequent is traditionally interpreted as a request to calculate a term or *answer substitution*, $t$, such that the

$$\frac{}{\Gamma\,;\,\varphi} \qquad (\xi{:}\varphi \in \Gamma) \qquad \text{L1}$$

$$\frac{\Gamma, \xi{:}\varphi\,;\,\psi}{\Gamma\,;\,\varphi \supset \psi} \qquad (\xi \notin \mathrm{dom}\Gamma) \qquad \text{L2}$$

$$\frac{\Gamma, \eta{:}\psi\,;\,\chi \quad \Gamma\,;\,\varphi}{\Gamma\,;\,\chi} \qquad (\xi{:}\varphi \supset \psi \in \Gamma) \qquad \text{L3}$$

Figure 1: L1 – L3 displayed.

sequent $\Gamma; \psi[t/x]$ is provable. In order to determine that such a situation obtains, it is necessary to calculate both a term $t$ and a proof of the sequent $\Gamma; \psi[t/x]$.

With the aforesaid understanding in mind, and using $l, k$, possibly subscripted, to denote sequents, we define

$$N(\Gamma; \varphi) \ := \ \left\{M \in \mathcal{N} \mid \Gamma \models^{M} \varphi\right\}$$

and

$$N_0(l) \ := \ \{M \in N(l) \mid M \,\text{normal}\},$$

where "normal" here is taken in the sense of Prawitz [14].

A (cut-free) sequent calculus ($L$) for a pure implicational logic is defined as follows: for each sequent $l$ consider a family of pairwise distinct variables $(\alpha^{l}_{m})_{m<\omega}$. These variables are used to represent the unproved or *open* leaves of a derivation. The set of derivations of sequent $l$, denoted $D(l)$, is the smallest set that contains the variables and is closed under the following rules:

L1. $\text{AXIOM}_{\Gamma,\varphi}(\xi) \in D(\Gamma; \varphi)$ if $\xi{:}\varphi \in \Gamma$;

L2. $\text{IMP-R}_{\Gamma,\varphi,\psi}(\xi, d) \in D(\Gamma; \varphi \supset \psi)$ if $d \in D(\Gamma, \xi{:}\varphi; \psi)$ and $\xi \notin \mathrm{dom}\Gamma$;

L3. $\text{IMP-L}_{\Gamma,\varphi,\psi,\chi}(\eta, \xi, d, d') \in D(\Gamma; \chi)$ if $d \in D(\Gamma; \varphi)$, $d' \in D(\Gamma, \eta{:}\psi; \chi)$, $\eta \notin \mathrm{dom}\Gamma$ and $\xi{:}\varphi \supset \psi \in \Gamma$.

If $d \in D(l)$ we often write $d^{l}$.

In terms of our sequential understanding of logic programming, clauses L1 – L3, displayed as rules in Figure 1, should be considered to determine *reduction operators* — rules that are read from conclusion to premisses.[3] With this reading, SLD-resolution can be considered to amount to taking a specialised form of L3, namely $L_{\iota}3$, *q.v.* § 6.

[3] Kleene [6] explains this in the setting of the classical predicate calculus.

## 4 Completions and extensional equivalence

The traditional interpretation of the sequent calculus is that it builds normal natural deductions. In an (intuitionistic) first-order setting, such deductions contain explicit witnesses for existential quantifiers.

Let $V(d)$ denote the set of variables of a derivation $d$. A mapping that assigns a natural deduction $M \in N(l)$ to each variable $\alpha^l$, is called a *completion*[4].

Let $\rho$ be a completion. The *extensional* interpretation relative to $\rho$, denoted $[\![.]\!]^E_\rho$, is defined by recursion on the structure of sequent derivations as follows:

E0. $[\![\alpha]\!]^E_\rho = \rho(\alpha)$;

E1. $[\![\text{AXIOM}_{\Gamma,\varphi}(\xi)]\!]^E_\rho = \text{HYP}_\varphi(\xi)$;

E2. $[\![\text{IMP-R}_{\Gamma,\varphi,\psi}(\xi, d)]\!]^E_\rho = \text{IMP-I}_{\varphi,\psi}(\xi : [\![d]\!]^E_\rho)$;

E3. $[\![\text{IMP-L}_{\Gamma,\varphi,\psi,\chi}(\eta, \xi, d, d')]\!]^E_\rho =$

$$([\![d']\!]^E_\rho)[\text{IMP-E}_{\varphi,\psi}(\text{HYP}_{\varphi\supset\psi}(\xi), [\![d]\!]^E_\rho)/\text{HYP}_\psi(\eta)].$$

4.1 PROPOSITION *For any completion $\rho$, if $d \in D(l)$ then $[\![d]\!]^E_\rho \in N(l)$.*

Define the relation of *extensional equivalence*, $\sim$, by

$$d \sim d' \qquad \text{iff} \qquad (\forall\rho)\, [\![d]\!]^E_\rho = [\![d']\!]^E_\rho.$$

Two extensionally equivalent derivations are shown in Figure 2; they are permutation variants of each other [5].

## 5 Constructions

The interpretation $[\![.]\!]^E_\rho$ above is suitable for interpreting the *results* of computations (searches) via the sequent calculus, *i.e.*, for interpreting sequent *proofs*. However, since the two derivations of Figure 2 require differing amounts of work to complete — assuming such completions exist — they represent clearly distinct *partial* computations. In this section we introduce an interpretation of derivations that captures this difference and is therefore suitable for interpreting states of computations (*i.e.*, partial derivations). The idea is very simple: we internalise the operation of substitution $[M^\varphi/\text{HYP}_\varphi(\xi)]$ rather than treating it as a metatheoretic operation.

The set of *constructions* $C$ is defined to be the smallest set containing $\text{HYP}_\varphi(\xi)$, $\text{IMP-I}_{\varphi,\psi}(\xi : t)$, $\text{IMP-E}_{\varphi,\psi}(t, s)$ and $\text{SUB}(t, s/\text{HYP}_\varphi(\xi))$ whenever $\varphi, \psi \in \mathcal{F}$, $\xi$ is a marker and $t, s \in C$. End formulae of constructions are defined as for proof

[4]This notion of completion is distinct from that of Clark [8].

$$\dfrac{\dfrac{\overline{\Gamma,\xi:\psi\supset\chi,\eta:\chi;\chi} \qquad \dfrac{\overline{\Gamma,\xi:\psi\supset\chi,\zeta:\psi;\psi} \quad \Gamma,\xi:\psi\supset\chi;\varphi}{\Gamma,\xi:\psi\supset\chi;\psi}}{\Gamma,\xi:\psi\supset\chi;\chi} \qquad \dfrac{\overline{\Gamma,\zeta:\psi;\psi} \quad \Gamma;\varphi}{\Gamma;\psi}}{\Gamma;\chi}$$

$$\dfrac{\dfrac{\dfrac{\overline{\Gamma,\xi:\psi\supset\chi,\zeta:\psi,\eta:\chi;\chi} \quad \overline{\Gamma,\xi:\psi\supset\chi,\zeta:\psi;\psi}}{\Gamma,\xi:\psi\supset\chi,\zeta:\psi;\chi} \qquad \overline{\Gamma,\zeta:\psi;\psi}}{\Gamma,\zeta:\psi;\chi} \qquad \Gamma;\varphi}{\Gamma;\chi}$$

$$\Gamma=\{\xi_1:\varphi\supset\psi,\xi_2:\psi\supset(\psi\supset\chi)\}$$

Figure 2: Two extensionally equivalent derivations.

expressions with the addition that the end formula of a construction of the form $\mathrm{SUB}(t,s/\mathrm{HYP}_\varphi(\xi))$ is the end formula of $t$. The intended meaning of this construct is that construction $s$ is substituted for (all) free occurrences of $\mathrm{HYP}_\varphi(\xi)$ in $t$. It is clear that $\mathcal{N}\subset\mathcal{C}$.

The set of well-formed context-construction pairs $(\Delta,t)$ is the smallest set satisfying:

C1. $(\Delta,\mathrm{HYP}_\varphi(\xi))$ is well-formed whenever $\Delta$ is well-formed and $\xi:\varphi\in\Delta$;

C2. $(\Delta,\mathrm{IMP\text{-}I}_{\varphi,\psi}(\xi:t)^{\varphi\supset\psi})$ is well-formed whenever $(\Delta,\xi:\varphi\ ,\ t^\psi)$ is well-formed;

C3. $(\Delta,\mathrm{IMP\text{-}E}_{\varphi,\psi}(t,s)^\psi)$ is well-formed whenever $(\Delta,t^{\varphi\supset\psi})$ and $(\Delta,s^\varphi)$ are well-formed;

C4. $(\Delta,\mathrm{SUB}(t,s^\varphi/\mathrm{HYP}_\varphi(\xi)))$ is well-formed whenever $(\Delta,t)$ and $(\Delta,s)$ are well-formed.

We write $\Delta\models^t\varphi$ if $(\Delta,t^\varphi)$ is well-formed.

Define

$$\mathcal{C}(\Gamma;\varphi)\ :=\ \left\{t\in\mathcal{C}\mid\Gamma\models^t\varphi\right\}.$$

Let $\rightarrow$ denote the reduction relation for substitution over constructions in the usual manner. Call a construction *explicit* if it contains no substitution redices. $\rightarrow$ is Church-Rosser and strongly normalising for well-formed constructions. Consequently the explicit form of a construction is unique. Substitution preserves well-formedness, and explicit constructions are normal deductions.

5.1 Proposition *If $t \in C(l)$, then:*

*(i) If $t \to s$ then $s \in C(l)$;*

*(ii) If $s$ is the explicit form of $t$, then $s \in N_0(l)$.*

The *constructional* interpretation relative to a completion $\rho$, denoted $[\![.]\!]^S_\rho$, is defined inductively on the structure of sequent derivations as follows:

S0. $[\![\alpha]\!]^S_\rho = \rho(\alpha)$;

S1. $[\![\text{AXIOM}_{\Gamma,\varphi}(\xi)]\!]^S_\rho = \text{HYP}_\varphi(\xi)$;

S2. $[\![\text{IMP-R}_{\Gamma,\varphi,\psi}(\xi, d)]\!]^S_\rho = \text{IMP-I}_{\varphi,\psi}(\xi : [\![d]\!]^S_\rho)$;

S3. $[\![\text{IMP-L}_{\Gamma,\varphi,\psi,\chi}(\eta, \xi, d, d')]\!]^S_\rho =$

$$\text{SUB}([\![d']\!]^S_\rho, \text{IMP-E}_{\varphi,\psi}(\text{HYP}_{\varphi\supset\psi}(\xi), [\![d]\!]^S_\rho)/\text{HYP}_\psi(\eta)).$$

The counterpart to Proposition 4.1 is:

5.2 Proposition *For any completion $\rho$, if $d \in D(l)$ then $[\![d]\!]^S_\rho \in C(l)$.*

## 6 Some results

Proposition 5.1(ii) tells us that the substitution reduction $\to$ embeds the set of constructions $C(l)$ into the set of normal natural deductions $N_0(l)$. (In fact the embedding is surjective.) As expected, $[\![.]\!]^E$ factors through $[\![.]\!]^S$ via reduction to explicit form using $\to^*$, the transitive closure of $\to$, *i.e.*,

$$[\![.]\!]^E \quad = \quad (\to^* \circ \quad [\![.]\!]^S).$$

In fact we can pick out a copy of $C(l)$ in $N(l)$.

Define a mapping $\tau: \mathcal{C} \to \mathcal{N}$ by recursion on the structure of constructions as follows:

T1. $\tau(\text{HYP}_\varphi(\xi)) = \text{HYP}_\varphi(\xi)$;

T2. $\tau(\text{IMP-I}_{\varphi,\psi}(\xi : t)) = \text{IMP-I}_{\varphi,\psi}(\xi : \tau(t))$;

T3. $\tau(\text{IMP-E}_{\varphi,\psi}(t, s)) = \text{IMP-E}_{\varphi,\psi}(\tau(t), \tau(s))$;

T4. $\tau(\text{SUB}(t^\chi, s/\text{HYP}_\varphi(\xi))) = \text{IMP-E}_{\psi,\chi}(\text{IMP-I}_{\psi,\chi}(\xi : \tau(t))\ ,\ \tau(s))$.

It is perhaps easier to see the effect of this mapping pictorially in Figure 3, which illustrates an instance of T4: given a construction $t$ of $\chi$ from assumption $\psi$, we can replace occurrences of the marker $\xi$, standing for occurrences of a (hypothetical) construction of $\psi$, by a construction $s$ of $\psi$: the resulting construction, represented on the left of the $\mapsto$ in Figure 3, corresponds to the non-normal natural deduction tree on the right of the $\mapsto$ in Figure 3. Indeed, this situation is characteristic of T4, which distinguishes constructions from (normal) natural deductions as follows:

$$\text{SUB}\left(\begin{array}{c}\xi:\psi\\ \vdots\\ \chi\end{array} t\ ,\ \begin{array}{c}\vdots\\ \psi\end{array} s\ /\ (\xi:\psi)\right) \ \stackrel{\tau}{\longmapsto}\ \dfrac{\dfrac{\begin{array}{c}[\xi:\psi]\\ \vdots\ \tau(t)\\ \chi\end{array}}{\psi\supset\chi} \qquad \begin{array}{c}\vdots\ \tau(s)\\ \psi\end{array}}{\chi}$$

Figure 3: Identification of construction with non-normal deduction.

6.1 PROPOSITION $\tau[\,C(l)\setminus N(l)\,]\cap N_0(l)=\emptyset$.

Although $\tau$ maps true constructions to non-normal deductions, there is a class of non-normal deductions that are essentially normal. *Identity* reduction, $\rightarrow_\iota$, is a notion of reduction defined on proof expressions as follows:

$$\text{IMP-E}_{\psi,\psi}(\text{IMP-I}_{\psi,\psi}(\xi:\text{HYP}_\psi(\xi))\ ,\ M)\ \rightarrow_\iota\ M.$$

This is a special case of normalisation [14]. Notice that the redex on the left is a particular case of the image under $\tau$ of a substitution (see T4 above). Let $=_\iota$ denote the congruence generated by $\rightarrow_\iota$ in the usual manner. Call $M\in N(l)$ *$\iota$-normal* if it is normal or $\iota$-congruent to a normal deduction. Let $N_\iota(l)$ denote the set of $\iota$-normal deductions of $l$.

Let $L_\iota$ denote the sequent calculus formed by replacing the IMP-L rule of $L$ by the rule below; *cf.* §3. ($D_\iota(l)$ denotes the derivations of $l$ in $L_\iota$.)

$L_\iota3$. $\text{IMP-L}_{\Gamma,\varphi,\chi,\chi}(\eta,\xi,d,\text{AXIOM}_{(\Gamma,\eta:\chi),\chi}(\eta))\in D_\iota(\Gamma;\chi)$ if $d\in D_\iota(\Gamma;\varphi)$, $\eta\notin\text{dom}\Gamma$ and $\xi:\varphi\supset\chi\in\Gamma$.

$L_\iota3$ is a derived rule of $L$; *i.e.*, $D_\iota(l)\subset D(l)$. L3 is only admissible in $L_\iota$. These rules are displayed in Figure 4. Notice that both rules make use of implicit contraction.

A completion is said to be *$\iota$-normal* (resp. *normal*) if, for all $\alpha^l$, $\rho(\alpha)\in N_\iota(l)$ (resp. $N_0(l)$).

6.2 PROPOSITION *(i)* $(\tau\circ[\![.]\!]^S_\rho)[D_\iota(l)]=N_\iota(l)$, *for all $\iota$-normal $\rho$.*

*(ii)* $(\tau\circ[\![.]\!]^S_\rho)[D(l)]\supset N_\iota(l)$, *for all $\iota$-normal $\rho$.*

Computations (searches) using the system $L_\iota$ involve only $\iota$-normal proofs whereas computations using the system $L$ encounter a wider class of deduction. To this extent $\iota$-normal proofs characterise the constructions that result from (an analytic account of) computations via SLD-resolution. Despite the fact that $\iota$-normal proofs need not be normal, reduction to normal form can be performed on *partial* proofs and hence the residual computation can be interleaved with the primary computation.

| | | |
|---|---|---|
| IMP-L of $L$ | $\dfrac{\Gamma, \eta{:}\psi \,;\, \chi \quad \Gamma \,;\, \varphi}{\Gamma \,;\, \chi}$ | $\xi{:}\varphi \supset \psi \in \Gamma$ |
| IMP-L of $L_\iota$ | $\dfrac{\overline{\Gamma, \eta{:}\chi \,;\, \chi} \quad \Gamma \,;\, \varphi}{\Gamma \,;\, \chi}$ | $\xi{:}\varphi \supset \chi \in \Gamma$ |

Figure 4: The rules L3 and $L_\iota 3$ displayed.

$L$, on the other hand, embodies a form of limited (analytic) cut on *left* or *negative* subformulae. Thus the construction of a proof ($[\![.]\!]^S$) may involve less work than the explicit presentation ($[\![.]\!]^E$) of that proof; the difference being measured by the effect of the elimination of the substitution operator. However, the substitution constructs *cannot* be eliminated until certain subproofs have been fully completed (those with $\eta{:}\psi$ in their antecedents).

It is obvious, though we shall not develop it here, that the extension of the constructional interpretation $[\![.]\!]^S_\rho$ to first-order programs would yield non-trivial residual computations.

## References


[1] R.L. Constable *et al.*. Implementing mathematics with the NuPRL proof development system. Prentice Hall, 1986.

[2] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.,* 50(1), 1987.

[3] K. Gödel. Uber eine bisher noch nicht benützte Erweiterung des finiten Standpunktes, *Dialectica,* 12, 1958, pp.280–287; English translation, *J. Phil. Logic,* 9, 1980, pp.133–142.

[4] R. Harper, F. Honsell, G. Plotkin. A framework for defining logics. *J. ACM,* (to appear). (Preprint available as University of Edinburgh LFCS report ECS-LFCS-91-162, June 1991.)

[5] S.C. Kleene. Permutability of inferences in Gentzen's calculi **LK** and **LJ**. *Mem. Amer. Math. Soc.*, pages 1–26, 1952.

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

[7] R. Kowalski and D. Kuehner. Linear resolution with selection function. *J. Artif. Int.*, 2:227–260, 1971.

[8] J. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[9] D.W. Loveland. *Automated theorem proving: a logical basis.* Elsevier, 1979.

[10] P. Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. Technical Report 2, Scuola di Specializziazione in Logica Matematica, Dipartimento di Matematica, Università di Siena, 1985.

[11] D. Miller and G. Nadathur. Higher-order Logic Programming. Report MS-CIS-86-17, University of Pennsylvania, 1986.

[12] D. Miller, G. Nadathur, A. Ščedrov and F. Pfenning. Uniform Proofs as a Foundation for Logic Programming. *Ann. Pure and Appl. Logic,* 51, pp. 125–157, 1991.

[13] F. Pfenning. Logic programming in the LF logical framework. In: *Logical Frameworks*, G. Huet and G. Plotkin (editors), Cambridge University Press, 1991. pp. 149–181.

[14] D. Prawitz. *Natural deduction: a proof-theoretical study.* Almqvist & Wiksell, 1965.

[15] D. Prawitz. Proofs and the meaning and completeness of the logical constants. In: J. Hintikka, I. Niiniluoto and E. Saarinen (eds.) *Essays on Mathematical and Philosophical Logic*, pp. 25-40. D. Reidel, 1978.

[16] D.J. Pym. *Proofs, Search and Computation in General Logic.* Ph.D. thesis, University of Edinburgh, 1990. Available as report CST-69-90, Department of Computer Science, University of Edinburgh, 1990. (Also published as LFCS report ECS-LFCS-90-125.)

[17] D.J. Pym and L.A. Wallen. Proof-search in the $\lambda\Pi$-calculus. In: *Logical Frameworks*, G. Huet and G. Plotkin (editors), Cambridge University Press, 1991. pp. 309–340.

[18] W. Sieg. Herbrand analyses. *Arch. Math. Logic*, 30, pp. 409–441, 1991.

[19] J. Zucker. The correspondence between cut-elimination and normalisation. *Ann. Math. Logic*, 7:1–112, 1974.