Proof search in constructive logics *

Roy Dyckhoff
Computer Science Division, University of St Andrews, Scotland
email: rd@dcs.st-and.ac.uk

Luís Pinto
Departamento de Matemática, Universidade do Minho, Portugal
email: luis@math.uminho.pt

Abstract

We present an overview of some sequent calculi organised not for "theorem-proving" but for proof search, where the proofs themselves (and the avoidance of known proofs on backtracking) are objects of interest. The main calculus discussed is that of Herbelin [1994] for intuitionistic logic, which extends methods used in hereditary Harrop logic programming; we give a brief discussion of some similar calculi for other logics. We also point to some related work on permutations in intuitionistic Gentzen sequent calculi that clarifies the relationship between such calculi and natural deduction.

1 Introduction

It is widely held that ordinary logic programming is based on classical logic, with a Tarski-style semantics (answering questions "What judgments are provable?") rather than a Heyting-style semantics (answering questions like "What are the proofs, if any, of each judgment?"). If one adopts the latter style (equivalently, the BHK interpretation: see [35] for details) by regarding proofs as answers to questions, or as solutions to problems, then proof-enumeration rather than theorem-proving is the issue. See [12] for discussion of differences between the two styles of semantics.

*Both authors were supported by the European Commission via the ESPRIT BRA 7232 GENTZEN and by the Centro de Matemática da Universidade do Minho, Portugal.

Some authors (e.g. [25]) have shown that as an adequate basis for pure Prolog one can, instead of classical logic, take the Horn fragment of minimal logic, and that this can be extended up to the hereditary Harrop fragment of minimal (or, equivalently, intuitionistic) logic, thus providing [24] a good logical basis for software engineering features such as scoping and modularity. In such work, the emphasis is on provability: the semantics is Tarski-style rather than Heyting-style. The uniform proof system of [25], a Gentzen-style sequent calculus with side-conditions restricting the applicability of certain rules, is presented merely as an efficient mechanism for answering questions about provability rather than as a mechanism for enumerating proofs.

Our purpose in this paper is to argue that one should go further: that the semantics of such languages should be Heyting-style, and that the appropriate proof-search calculi are those that have not only the syntax-directed features of Gentzen-style sequent calculi but also a natural 1-1 correspondence between the derivations and the real objects of interest, normal natural deductions.

Such calculi are to be found in the work of Howard [18] and of Herbelin [15], [16]. Howard's idea (attributed to Curry) is close to the terminology of logic programming, but does not generalise beyond the $(\wedge, \supset, \forall)$-fragment of minimal logic. Herbelin's work was motivated by the application to functional programming, replacing the ordinary typed λ-calculus by a new calculus $\overline{\lambda}$ of terms representing derivations in a sequent calculus, where the cut-elimination rules form a clean and simple calculus of explicit substitutions. The same calculus (without the *Cut* rule) may, we observe, be used effectively as a proof-search calculus.

In this paper we look at such calculi for some constructive logics, where a logic is (loosely) regarded as *constructive* if the disjunction and existence properties hold for it. As is well-known, classical logic is not constructive in this sense: disjunctive formulae $p \vee \neg p$ are provable without either of the literals p and $\neg p$ being provable, and some existential formulae $\exists x.U(x)$ (such as $\exists x.\forall y.(px \supset py)$) are provable without $U(t)$ being provable for any particular t. Another aspect of constructivity that we consider is the emphasis on proofs (as "constructions") rather than just on provability. Extension of the ideas to (for example) classical logic is an interesting challenge, explored in [15]: one of the problems is that of agreeing on a suitable proof system to provide the "semantics". We choose not to discuss whether classical logic is "constructive" as argued in [13].

We consider therefore (first-order) minimal logic, intuitionistic logic, an intuitionistic modal logic and intuitionistic linear logic. Elsewhere [29] we report on work extending these ideas to a dependent type theory based on that of [32].

The calculi that we consider to be good candidates for proof search have derivations in 1-1 correspondence with the normal natural deductions. The

usual calculi of Gentzen and Kleene do have derivations interpretable as normal natural deductions, but the interpretation is many-one. We show elsewhere [5], in the case of one calculus (roughly that of Kleene) for intuitionistic logic, how the fibres of this interpretation are generated by the Kleene-style permutations. In other words, the permutabilities in (cut-free) LJ are an obstacle to the use of LJ as a proof-search calculus. For this reason, we have in [4, 5, 6] described the new proof-search calculi as"permutation-free": any "permutation" in such calculi would be dismissed as semantically unsound. Note that in contrast it is the non-permutabilities of a calculus that are obstacles ([36], p. 50) to its use for efficient automated theorem proving.

Further, in the case of disjunction, there are (see [5]) Kleene-style permutations in LJ that are not semantically sound, according to the usual equational theory of the typed lambda calculus. This is a second feature of LJ that makes it a poor proof search calculus.

Proofs of all results mentioned here appear elsewhere, mainly in [4, 5, 6, 19, 20, 29].

2 Background

Notations are as in [34], except that we use $A \supset B$ for implicational formulae. $\bot$ is a logical constant and not an atomic formula. $\neg A$ abbreviates $A \supset \bot$.

A formula is *Horn* when it is the closure of a formula of the form $B \supset H$, where the *head* H is atomic and the *body* B is a (possibly trivial) conjunction of atoms; when B is trivial we replace it by $true$, and $true \supset H$ we replace by H. A formula is *hereditary Harrop* (resp., an *hH goal*) when every occurrence therein of either a disjunction or an existential subformula is negative (resp., positive): cf. [24, 25] for details, but note that the definition therein differs inessentially from ours by prohibiting formulae such as $p_1 \supset (p_2 \wedge p_3)$ and $\bot$ (and in being less memorable).

There are several ways [27] to extend a pure logical calculus with a theory given by axioms. One way [12, 14], assuming that the axioms are Horn formulae, is to interpret them as new inference rules, as reflected in the traditional logic programming term "rules". We adopt instead the approach of Gentzen, where a *theory* is a list Δ of closed formulae: the *achievement* of a *goal* G (where G is any closed formula) w.r.t. such a theory is just a proof of the sequent $\Delta \Rightarrow G$. We shall also call a theory a *logic program*. The (Heyting-style) semantics of a logic program Δ is then just the association to each goal G of the set of proofs of $\Delta \Rightarrow G$.

As the proofs, we consider as primary the normal natural deductions, for the simple reasons that (i) (under the Curry-Howard correspondence) such proofs correspond to values in the sense of functional programming, (ii)

natural deductions are well understood and (iii) they lack the redundancy of traditional Gentzen-style sequent calculi (arising from the permutations [21] therein). We consider also the expanded normal deductions of [30].

"Normal" is defined as in [34]. Note that this restricts application of $\bot E$ to cases where the conclusion is atomic. $\bot$ in fact causes many problems. Our guiding principle here is that there should be exactly one normal natural deduction proof of $\top =_{def} \bot \supset \bot$; there are two possible candidates:

$$\frac{[\bot]}{\bot \supset \bot} \supset I \qquad \frac{\dfrac{[\bot]}{\bot} \bot E}{\bot \supset \bot} \supset I$$

of which we choose the first, both for simplicity and because $\bot$ is not an atom and so we may reject the second. A consequence of this choice is that if we try to restrict sequent calculus axioms $\Delta, A \Rightarrow A$ in LJ to atomic A, then we have also to allow their use when $A = \bot$.

The cut-free calculus LJ of Gentzen [11] (hereafter "G1i", as in [34]) is a starting point for automated proof search: proof search in G1i is "syntax-directed", in the sense that active formulae are always the same as, or immediate subformulae of, the principal formula, and G1i derivations can be interpreted as normal NJ proofs. The variations G2i and G3i, incorporating the structural rules into the logical rules, are even better [34] for this purpose. However, the permutations mentioned above impose an undesired redundancy on backtracking: the same NJ proof may be rediscovered several, possibly infinitely many, times. See [5] for a discussion of these permutations, with permutation reductions and a normalisation argument (extended to strong normalisation for a related system in [33]). The present paper therefore looks at calculi without these permutability properties.

3 Herbelin's calculus

Motivated by interest in obtaining a calculus of explicit substitutions based on sequent calculus, Herbelin has introduced a new λ-calculus of terms and a type system for it that we can regard as a deduction system with proof terms. Rather than use his name "LJT" (already used for a system introduced in our [3]), we use another name: we call our modification of his system "MJ", since it is intermediate between LJ and NJ.

The *formulae* $A, B, ..., G, ...$ of MJ are as usual. The two categories M and Ms of *terms* are described by the two grammars

$$M ::= (V; Ms) \mid \lambda V.M \mid in_i(M) \mid pair(M, M) \mid \lambda W.M \mid pairq(T, M)$$
$$Ms ::= ax \mid ae \mid (M :: Ms) \mid when(V.M, V.M) \mid p_i(Ms) \mid apq(T, Ms) \mid spl(W.V.M)$$

in which $i = 1, 2$; V is the category of (proof) variables $x, y, \ldots$, W is the category of (individual) variables $u, v, w, \ldots$ and T is the usual category of terms $t, \ldots$ built from W by means of some function symbols. Variable binding occurs at occurrences of $V.M$, $W.M$ and $W.V.M$, with the usual conventions.

Contexts Δ are partial functions from V to formulae, written (in arbitrary order) as a sequence of *declarations* $x : A$. There are two forms of sequent: the forms $\Delta \Rightarrow M : G$ and $\Delta \stackrel{A}{\longrightarrow} Ms : G$, respectively called "ordinary sequents" and "stoup sequents". The position above the arrow in the second form of sequent is called the "stoup". The terms of the calculus admit a natural 1-1 onto translation to the terms of the typed lambda calculus (with types corresponding to the formulae of first-order logic). Details of this translation can be found in [6], including checks that it works not just at the level of terms but also of inference rules.

The *axioms* for this calculus are of the form $\Delta \stackrel{A}{\longrightarrow} ax : A$. The inference rules, in which R is for "Right", S is for "Stoup" and Sel is for "Selection" (from the context into the stoup), are as follows:

$$\frac{\Delta, x : A \stackrel{A}{\longrightarrow} Ms : B}{\Delta, x : A \Rightarrow (x; Ms) : B}\ Sel \qquad \frac{}{\Delta \stackrel{\perp}{\longrightarrow} ae : C}\ S\perp$$

$$\frac{\Delta, x : A \Rightarrow M : B}{\Delta \Rightarrow \lambda x.M : A \supset B}\ R\supset \qquad \frac{\Delta \Rightarrow M : A \quad \Delta \stackrel{B}{\longrightarrow} Ms : C}{\Delta \stackrel{A\supset B}{\longrightarrow} (M :: Ms) : C}\ S\supset$$

$$\frac{\Delta \Rightarrow M : A_i}{\Delta \Rightarrow in_i(M) : A_1 \vee A_2}\ R\vee_i \qquad \frac{\Delta, x : A \Rightarrow M : C \quad \Delta, y : B \Rightarrow M' : C}{\Delta \stackrel{A\vee B}{\longrightarrow} when(x.M, y.M') : C}\ S\vee$$

$$\frac{\Delta \Rightarrow M : A \quad \Delta \Rightarrow M' : B}{\Delta \Rightarrow pair(M, M') : A \wedge B}\ R\wedge \qquad \frac{\Delta \stackrel{A_i}{\longrightarrow} Ms : C}{\Delta \stackrel{A_1\wedge A_2}{\longrightarrow} p_i(Ms) : C}\ S\wedge_i$$

$$\frac{\Delta \Rightarrow M : U(w)}{\Delta \Rightarrow \lambda w.M : \forall U}\ R\forall \qquad \frac{\Delta \stackrel{U(t)}{\longrightarrow} Ms : C}{\Delta \stackrel{\forall U}{\longrightarrow} apq(t, Ms) : C}\ S\forall$$

$$\frac{\Delta \Rightarrow M : U(t)}{\Delta \Rightarrow pairq(t, M) : \exists U}\ R\exists \qquad \frac{\Delta, x : U(w) \Rightarrow M : C}{\Delta \stackrel{\exists U}{\longrightarrow} spl(w.x.M) : C}\ S\exists$$

in which the usual side conditions are imposed by the notation (e.g. that w is new in $R\forall$) and U stands for an abstraction of a formula w.r.t. an individual variable, so $U(t)$ is a formula for any term t. (U is for "unsaturated", as used by Frege.) Note that the Sel rule corresponds to the contraction rule of LJ.

In root-first proof search we shall say that a rule is *applicable* when the conclusion of one of its instances matches the current sequent.

4 Non-determinism in Herbelin's calculus

MJ has clear sources of non-determinism. Consider root-first search for a proof of a sequent $\Delta \Rightarrow M : G$ (where M is yet unknown). At most two rules are applicable: the *Sel* rule for selecting a member $x : A$ of the context (and copying it into the stoup), and the rule that introduces the principal connective of G, if any. Similarly, in searching for a proof of a sequent $\Delta \stackrel{A}{\longrightarrow} Ms : B$, where Ms is unknown, we only consider the formula A. If this is an implication or disjunction or an atom or absurdity ($\bot$) or existential, we have no choice; if a conjunction, then we must choose a conjunct; if a universal formula, then we must choose an instantiating term t. (This choice of t can be delayed, using the usual technique of unification proposed by Herbrand and developed by Prawitz [31].) Thus the only non-determinism is associated with the *Sel* rule, the $S\wedge$ rules and (less seriously) the $S\forall$ rule.

The use of stoup sequents is thus a form of *focusing*, as introduced by Andreoli [1]. However, note that some rules have an ordinary sequent as premise and a stoup-sequent as conclusion, thus allowing a transition back (during root-first proof search) to ordinary sequents and abandonment of the focus on the stoup formula. These rules are those dealing with (in the stoup) an implication, a disjunction or an existential formula.

The first (implication) is not too problematic, since the $S\supset$ rule involves (as we move from conclusion to left premise) a change in the goal. This is exactly what happens in logic programming when a program clause $B\supset H$ is selected, with H matching the current goal and the new goal being (an instantiation of) B. B is often in practice organised so that the search terminates. The matching of H with the goal corresponds to the right premise of $S\supset$ being an axiom.

The possibility of the stoup formula being a disjunction, however, is more inconvenient: there is no corresponding change of goal, and the effect is to add a new declaration $x : A$ to the context, followed by the selection of a declaration from the context, perhaps even that which led to the disjunctive stoup formula. Similar remarks apply to existential formulae.

5 Herbelin's calculus and logic programming

The main form of non-determinism that we can avoid is the alternation between stoup-sequents and ordinary sequents arising from the presence of disjunctions or existential formulae in the stoup. Such formula occurrences can be excluded by restrictions to hereditary Harrop formulae in the program and to hH goals as goals. Such restrictions are presented in [25] as a means to allow the search to be *goal-directed*, in the sense that the only rules that may

be applied when the goal is compound are right rules: but even without the goal-directedness there are good reasons, just discussed, for these restrictions.

One further way of reducing the non-determinism (for an ordinary sequent) is to (try to) require that Sel only applies when the succedent is atomic. We could consider what restrictions are required in order not to affect the derivability of sequents $\Delta \Rightarrow M : A$ (but allowing M to change): but the essence of our approach is to consider first what class of normal deductions we are interested in and then what sequent calculus like MJ corresponds to it in a bijective fashion.

To this end, we may further require that the minimal formulae (of normal natural deductions) are atomic (or $\bot$); in this case we have the expanded normal form deductions of Prawitz [30] (equivalently, proof terms in $\beta\eta$-long normal form).

If we restrict use of axioms in MJ to the matching of atomic formulae (or of $\bot$), and the use of $S\bot$ to cases where the goal formula is atomic, and syntactically restrict the programs Δ to hereditary Harrop formulae and goals G to hH goals, then we may restrict uses of Sel to cases with atomic succedent. We now impose these restrictions. Let P range over atomic formulae. The proof system MJ^r, with axioms $\Delta \xrightarrow{P} ax : P$ and $\Delta \xrightarrow{\bot} ax : \bot$, is then:

$$\frac{\Delta, x : A \xrightarrow{A} Ms : P}{\Delta, x : A \Rightarrow (x; Ms) : P}\ Sel \qquad \frac{}{\Delta \xrightarrow{\bot} ae : P}\ S\bot$$

$$\frac{\Delta, x : A \Rightarrow M : B}{\Delta \Rightarrow \lambda x.M : A \supset B}\ R\supset \qquad \frac{\Delta \Rightarrow M : A \quad \Delta \xrightarrow{B} Ms : P}{\Delta \xrightarrow{A \supset B} (M :: Ms) : P}\ S\supset$$

$$\frac{\Delta \Rightarrow M : A_i}{\Delta \Rightarrow in_i(M) : A_1 \vee A_2}\ R\vee_i$$

$$\frac{\Delta \Rightarrow M : A \quad \Delta \Rightarrow M' : B}{\Delta \Rightarrow pair(M, M') : A \wedge B}\ R\wedge \qquad \frac{\Delta \xrightarrow{A_i} Ms : P}{\Delta \xrightarrow{A_1 \wedge A_2} p_i(Ms) : P}\ S\wedge_i$$

$$\frac{\Delta \Rightarrow M : U(w)}{\Delta \Rightarrow \lambda w.M : \forall U}\ R\forall \qquad \frac{\Delta \xrightarrow{U(t)} Ms : P}{\Delta \xrightarrow{\forall U} apq(t, Ms) : P}\ S\forall$$

$$\frac{\Delta \Rightarrow M : U(t)}{\Delta \Rightarrow pairq(t, M) : \exists U}\ R\exists$$

We thus achieve (in another notation) a slight generalisation of the uniform proof system of [25]. Proofs of hH goal formulae in MJ^r naturally correspond, in a 1-1 fashion, to the expanded normal form deductions. As a corollary we get (for hereditary Harrop logic) a 1-1 correspondence [7] between uniform proofs with back-chaining [24] and expanded normal natural deductions.

Search for a proof of $\Delta \Rightarrow M : G$ in this calculus (M being unknown) is goal-directed: it decomposes the goal G until it is atomic (or $\bot$), then selects a declaration $x : A$ from the program Δ and copies A into the stoup. A may be compound: but following now always the rightmost branch (in the case that the stoup formula is an implication) will either reduce it to $\bot$ (in which case we apply the rule $S\bot$) or reduce it to an atom, whose non-matching with the atomic (or $\bot$) goal would force backtracking to the last choice point. Such a point will either be where we chose $S\wedge_1$ rather than $S\wedge_2$ (or *vice versa*) or where we selected wrongly from the program into the stoup. Some pre-processing can of course speed this up even more, e.g. by arranging that formulae $A\supset(B\supset C)$ are replaced by $(A \wedge B)\supset C$ and that quantifiers appear inside rather than outside conjunctions; our point however is that the essence of the uniform proof system is hidden by such optimisations and lies in the (restricted) version of Herbelin's calculus just given.

This description of the proof search assumes that we search depth-first, but with minor modifications it also applies to breadth-first search.

Note that where a goal is existential (as goals implicitly are in, say, Prolog) any proof of the goal, say $\exists x.U(x)$, will end with an $R\exists$ step with the proof term $pairq(t, M)$ from which the "answer substitution" of t for x is trivially extracted. A similar extraction may be done where there are several existential quantifiers. It is an implementational rather than a logical issue how much of the rest of the proof term is made apparent to the user.

6 Semantics

Implicit in the above explanation is the assumption that normal natural deductions give the semantics, in contrast to the usual views that the semantics is Tarski-style (using minimal Herbrand models) and that the automatic method of "resolution" is the best way to answer questions about provability. One may see the development of resolution as one way to automate certain kinds of reasoning efficiently: this has led both to a view that automated reasoning *was* resolution (with various strategies and parameter adjustments to obtain efficiency for hard problems) and to a view that resolution and logic programming were connected. Resolution being classically based (with its conversions to CNF and negation of the goal), the constructive nature of logic programming was hidden and thus ignored by many (but not [25]). However, the operational semantics incorporated in Prolog interpreters actually gives answers with multiplicities that reflect the above proof system rather than the Tarski-style semantics. (Note that our system doesn't capture the order of solutions: nor does it capture the more subtle aspects of depth-first search: see [8] for solutions to this kind of problem.)

For example, the restricted version MJ^r of MJ provides answers not just about what but how many times and why, just as natural deduction does. Consider the classical problem "Is there a member of the list $[1,2,1,2]$?" in the context of the usual definition Δ (below) of *member* as a predicate relating items and lists. Classically, the answer is "Yes": a more detailed answer, in the classical version of constructivity, would be "Yes: 1 and 2." Even more detailed would be the answer "Yes, the members are 1 (twice) and 2 (twice)." Better still (but we don't achieve this in MJ^r) would be "Yes: the members are 1, 2, 1 (again) and 2 (again), in that order." MJ^r actually has the terms

$pairq(1,(m_1;apq(1,apq([2,1,2],ax)))).$
$pairq(2,(m_2;apq(2,apq([2,1,2],apq(1,(m_1;apq(2,apq([1,2],ax))::ax)))))).$
$pairq(1,....).$
$pairq(2,....).$

as proof-terms M for which $\Delta \Rightarrow M : \exists x.member(x,[1,2,1,2])$ is derivable, where Δ is the program

$$m_1 : \forall y.\forall z.member(y,cons(y,z)).$$
$$m_2 : \forall y.\forall z.\forall w.(member(y,z) \supset member(y,cons(w,z))).$$

and $[1,2,1,2]$ abbreviates $cons(1,cons(2,cons(1,cons(2,nil))))$ as usual. (The reader is invited to work out the missing terms and to apply the translation [6] into the standard natural deduction or lambda calculus terminology.)

The point here is not that it is useful to have the four proof terms, the traces of the computations, in full detail, but just that they are different. (However, extension of these ideas [29] to a more complex logic based on dependent type theory takes more account of the actual values.) Nor is it our point that the notation of MJ is the best way of presenting the terms: it is not as familiar as ordinary lambda notation, for example. We consider however that it is proof-theoretically attractive, to incorporate restrictions into the rules rather than into strategies for using the rules, and thus to have the calculus as an explicit search calculus.

7 Proof search in lax logic

Curry introduced in 1952 an intuitionistic modal logic—lax logic (LL)—recently rediscovered by several authors, with applications in hardware design [9, 23], constraint logic programming [10]; it is also [2] the type system CL of Moggi's computational lambda calculus [26]. It is essentially intuitionistic logic with a modal operator $\circ$ (read "somehow"), axiomatised by the axioms $A \supset (\circ A)$, $(\circ \circ A) \supset (\circ A)$ and $(A \supset B) \supset ((\circ A) \supset (\circ B))$. It may easily be

formalised as a Hilbert-style system, as a natural deduction system and as a Gentzen-style sequent calculus.

Howe [19] has extended the "permutation-free" approach described above to zero-order lax logic, allowing effective search for normal natural deductions. The new rules are just

$$\frac{\Delta \Rightarrow M : A}{\Delta \Rightarrow smhr(M) : \circ A}\ R\circ \qquad \frac{\Delta, x : A \Rightarrow M : \circ B}{\Delta \stackrel{\circ A}{\longrightarrow} smhl(x.M) : \circ B}\ S\circ$$

Extension [20] to the first-order case is routine. There are alternative approaches to the problem of effective problem solving in LL; one open research problem is the correct application of the "contraction-free" techniques from [3] at least to decide solvability of zero-order problems in LL. We consider however that the "permutation-free" approach, finding "all" proofs and not just deciding solvability, is the right way forward, being closer both to the logic programming motivation and to the computational lambda-calculus concern with proofs as terms.

8 Proof search in intuitionistic linear logic

Intuitionistic linear logic (ILL) is another constructive logic, with applications in logic programming [17]. Howe [20] has developed a (cut-free) sequent calculus for all of ILL with its derivations in 1-1 correspondence with the normal natural deductions of ILL, equivalent, on the fragment of ILL considered in [17], to the calculus therein.

9 Proof search in dependent type theory

Our interest in these proof search problems arises not only from ordinary logic programming but also from the desire to automate proof search in dependent type theory. It began with work [28] on the integration of functional and logic programming using a type-theoretic perspective, where both proof search and function evaluation are seen as two aspects of the same issue: the search for normal proofs (or normal lambda terms). It is our opinion that a proper integration of the two paradigms is best done by restriction to logically pure fragments of each, seen in each case as a fragment of type theory [22].

[29] presents a sequent calculus for typing the normal terms of the $\lambda\Pi$-calculus of [32], with an extension for the $\lambda\Pi\Sigma$-calculus, i.e. the extension of the $\lambda\Pi$-calculus with Σ-types. These sequent calculi have the same property as that explored above for MJ, admitting no permutations of the order in which rules are used. No clausal forms are required, in contrast to the resolution calculi of [32].

10 Acknowledgments

We thank Andrew Adams, Jacob Howe, Dale Miller, Sara Negri, Christian Urban and Jan von Plato for useful discussions and collaboration, and the authors of [8, 10, 15, 27, 33] for making them available before publication.

References

[1] J.-M. Andreoli. *Logic programming with focusing proofs in linear logic*, Journal of Logic and Computation, vol. 2, pp 297–347, 1992.

[2] N. Benton, G. M. Bierman, V. de Paiva. *Computational types from a logical perspective*, Technical Report TR-365, Computer Laboratory, University of Cambridge, 1995.

[3] R. Dyckhoff. *Contraction-free sequent calculi for intuitionistic logic*, Journal of Symbolic Logic, vol. 57, no. 3, pp 795–807, 1992.

[4] R. Dyckhoff, L. Pinto. *Cut-elimination and a permutation-free sequent calculus for intuitionistic logic*, Studia Logica, vol. 60, pp 107–118, 1998.

[5] R. Dyckhoff, L. Pinto. *Permutability of proofs in intuitionistic sequent calculi*, Theoretical Computer Science, to appear.

[6] R. Dyckhoff, L. Pinto. *A permutation-free sequent calculus for intuitionistic logic*, Research Report CS/96/9, Computer Science Division, St Andrews University, 1996, available from "`http://www-theory.dcs.st-and.ac.uk/~rd/`".

[7] R. Dyckhoff, L. Pinto. Uniform proofs and natural deductions, in: D. Galmiche and L. Wallen, eds., *Proceedings of CADE-12 workshop on "Proof search in type theoretic languages"* (Nancy, 1994) 17–23.

[8] B. Elbl. *A declarative semantics for depth-first logic programs*, Journal of Logic Programming, to appear.

[9] M. Fairtlough, M. Mendler. *An intuitionistic modal logic with applications to the formal verification of hardware*, Proceedings of the 1994 Conference on Computer Science Logic, Springer LNCS 933, pp 354–368, 1995.

[10] M. Fairtlough, M. Mendler, M. Walton. *First-order lax logic as a framework for constraint logic programming*, Technical Report MIPS-9714, Department of Mathematics and Computer Science, University of Passau, 1997.

[11] G. Gentzen. *The collected papers of Gerhard Gentzen* (ed. M. Szabo, North-Holland, Amsterdam, 1969).

[12] J.-Y. Girard, Y. Lafont, P. Taylor. *Proofs and Types*, Cambridge University Press, 1987.

[13] J.-Y. Girard. *A new constructive logic: classical logic*, Math. Structures Comput. Sci., vol. 1, pp 255–296, 1991.

[14] L. Hallnäs, P. Schroeder-Heister. *A proof-theoretic approach to logic programming. I. Clauses as rules*, Journal of Logic & Computation, vol. 1(2), pp 261–283, 1990.

[15] H. Herbelin. *A λ-calculus structure isomorphic to sequent calculus structure*, preprint (October 1994); available from "`http://capella.ibp.fr/~herbelin/LAMBDA-BAR-FULL.dvi.gz`".

[16] H. Herbelin. *A λ-calculus structure isomorphic to Gentzen-style sequent calculus structure*, Proceedings of the 1994 Conference on Computer Science Logic, Springer LNCS 933, pp 61–75, 1995.

[17] J. Hodas, D. Miller. *Logic programming in a fragment of intuitionistc linear logic*, Information and Computation, vol. 110, pp 327–365, 1994.

[18] W. A. Howard. *The formulae-as-types notion of construction*, in: "To H. B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism", (edited by J. R. Hindley and J. P. Seldin), Academic Press (1980).

[19] J. Howe. *A permutation-free calculus for lax logic*, Research Report CS/98/1, University of St Andrews, 1998.

[20] J. Howe. *Proof enumeration issues for intuitionist linear logic*, PhD thesis (in preparation), University of St Andrews, 1998.

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

[22] P. Martin-Löf. "Intuitionistic type theory", Bibliopolis, Naples, 1984.

[23] M. Mendler. *A timing refinement of intuitionistic proofs and its application to the timing analysis of combinational circuits*, Theorem Proving with Analytic Tableaux and Related Methods, Springer LNCS 1071, pp 261–277, 1996.

[24] D. Miller. *Abstractions in logic programs*, in: P. Odifreddi, ed., *Logic and computer science*, Academic Press, pp 329–359, 1990.

[25] D. Miller, G. Nadathur, F. Pfenning, A. Scedrov. *Uniform proofs as a foundation for logic programming*, Annals of Pure and Applied Logic, vol. 51, pp 125–157, 1991.

[26] E. Moggi. *Notions of computation and monads*, Information and Computation, vol. 93, pp 55–92, 1991.

[27] S. Negri, J. von Plato. *Cut elimination in the presence of axioms*, submitted, 1998.

[28] L. Pinto. *Proof-theoretic investigations into integrated logical and functional programming*, PhD thesis, University of St Andrews, 1996.

[29] L. Pinto, R. Dyckhoff. *Sequent calculi for the normal terms of the* $\lambda\Pi$- *and* $\lambda\Pi\Sigma$-*calculus*, Proceedings of CADE-15 workshop on Proof Search in Type Theoretic Languages, ed. D. Galmiche et al, CNRS, University of Nancy I, 1998.

[30] D. Prawitz. *Ideas and results in proof theory*, in: J. E. Fenstad, Proceedings of the Second Scandinavian logic symposium, North-Holland, pp 235–308, 1971.

[31] D. Prawitz. *An improved proof procedure*, Theoria, vol. 26, pp 102–139, 1960.

[32] D. Pym and L. Wallen. *Proof search in the* $\lambda\Pi$-*calculus*, in: G. Huet and G. Plotkin, eds., *Logical frameworks*, Cambridge University Press, pp 309–340, 1991

[33] H. Schwichtenberg. *Termination of permutative conversions in intuitionistic Gentzen calculi*, Theoretical Computer Science, to appear.

[34] A. S. Troelstra, H. Schwichtenberg. "Basic proof theory", Cambridge University Press, 1996.

[35] A. S. Troelstra, D. van Dalen. "Constructivism in mathematics, vol. 1", North-Holland, 1988.

[36] L. Wallen. "Automated deduction in non-classical logics", MIT Press, 1989.