Notes Towards a Semantics for Proof-search

David J. Pym

Department of Computer Science
Queen Mary and Westfield College
University of London
England, U.K.

Abstract
Algorithmic proof-search is an essential enabling technology throughout informatics. Proof-search is the proof-theoretic realization of the formulation of logic not as a theory of deduction but rather as a theory of reduction. Whilst deductive logics typically have a well-developed semantics of proofs, reductive logics are typically well-understood only operationally. Each deductive system can, typically, be read as a corresponding reductive system. We discuss some of the problems which must be addressed in order to provide a semantics of proof-searches of comparable value to the corresponding semantics of proofs. Just as the semantics of proofs is intimately related to the model theory of the underlying logic, so too should be the semantics of proof-searches. We discuss how to solve the problem of providing a semantics for proof-searches which adequately models both operational and logical aspects of the reductive system.

1 Introduction

Algorithmic proof-search is an essential enabling technology throughout informatics. Proof-search is the proof-theoretic realization of the formulation of logic not as a theory of deduction but rather as a theory of reduction. Whilst deductive logics typically have a well-developed semantics of proofs, reductive logics are typically well-understood only operationally. Each deductive system can, typically, be read as a corresponding reductive system. We discuss some of the problems which must be addressed in order to provide a semantics of proof-searches of comparable value to the corresponding semantics of proofs. Just as the semantics of proofs is intimately related to the model theory of the underlying logic, so too should be the semantics of proof-searches. We

1 The author is grateful for the partial support of the EPSRC via an Advanced Fellowship. He is grateful to Didier Galmiche for his invitation to write these notes for this volume.
2 Email: pym@dcs.qmw.ac.uk

© 2000 Published by Elsevier Science B. V. Open access under CC BY-NC-ND license.
discuss how to solve the problem of providing a semantics for proof-searches which adequately models both operational and logical aspects of the reductive system.

Although our focus here is on proof-search and its semantics, its counterpart in truth-functional semantics, \textit{i.e.}, model checking, provides an alternative perspective on reductive logic. We are certainly concerned to ensure that we develop a theory which is a good fit with model checking. However, discussion of these aspects is beyond our present scope and will be addressed elsewhere.

These notes are in many ways incomplete. In particular, we have, for brevity, refrained from providing any substantial detail of the semantic structures considered. Moreover, we do not claim to have fully investigated the lines of research herein. Rather, we sketch some aspects of our current research.

In § 2, we introduce the proof-theoretic view of reductive logic. In § 3, we discuss the formulation and rôle of the semantics of proofs and consider how the logical aspects of a similar semantics of proof-searches might be formulated. In § 4, we consider how to incorporate into the semantics an interpretation of a deterministic operational semantics. Finally, in § 6, we summarize our position and sketch our programme of research towards a semantics for proof-search.

2 Proof and proof-search

Deductive inference proceeds from established or supposed premisses to a conclusion, regulated by the application of inference rules, $R$,

$$
\downarrow \frac{\text{Premiss}_1 \ldots \text{Premiss}_m}{\text{Conclusion}} \quad R.
$$

A proof is constructed, inductively, by applying instances of rules of this form to proofs of established premisses, thereby constructing a proof of the given conclusion.

Reductive inference proceeds from a putative conclusion to sufficient premisses, regulated by reduction operators, $O_R$,

$$
\uparrow \frac{\text{SufficientPremiss}_1 \ldots \text{SufficientPremiss}_m}{\text{PutativeConclusion}} \quad O_R.
$$

corresponding to (admissible) rules, $R$. The author believes that this idea of reduction was first explained \textit{in these terms} by Kleene [18]. A search is constructed, inductively, by applying instances of reduction operators of this form to putative conclusions of which a proof is desired, thereby yielding a collection of sufficient premisses, proofs of which would be sufficient to imply the existence of a proof, obtainable by deduction, of the putative conclusion.
The key difference between reduction and deduction is that deduction are always \textit{total}, \textit{i.e.}, whenever a rule is applicable to a complete set of established premisses, then the conclusion of the rules can always be obtained, whereas reductions are, in general, \textit{partial}, \textit{i.e.}, when a reduction operator is applied to a putative conclusion, there is no guarantee that all, or indeed any, of the sufficient premisses will be provable. The leaves of a search are not necessarily axioms, $\phi \vdash \phi$, where $\phi$ is an atomic formula, of the underlying logic. Rather, they are a more general class of atomic judgements, $\phi \vdash \psi$, where $\phi$ and $\psi$ are atomic.

For the purposes of this paper, we shall assume that the proofs, and the searches, in a logic, are structured as trees and that such trees can be represented as the terms of a language of proofs. For example, the proofs of intuitionistic logic can be represented as trees generated by Gentzen’s natural deduction system NJ or sequent calculus LJ [7]. We emphasize, however, our conjecture that our analysis extends to a range of other graphical structures without undue difficulty.

We conclude by remarking that one of the leading examples of reductive proof, namely classical clausal \textit{resolution} [31], is often rendered in deductive form. The trick is to formulate the search for a proof of a formula as an attempt to deduce a contradiction from the negation of the formula [34]. Failure then arises as the failure of any instance of resolution to be applicable before a contradiction has been obtained. We suggest that a reductive formulation is more illuminating.

3 Semantics

What good is a semantics? What is a good semantics? Of course, the answers to these questions are much the same. For deductive logic, the key point is that the semantics should provide not only a more “abstract” interpretation of the syntax of the logic, including its proofs, but also an account of the meaning of the constructs of the logic, typically via an account of which judgments hold, \textit{e.g.}, are “true”, in which states of affairs, \textit{e.g.}, worlds.

The requirements for a semantics of deductive logic also obtain for reductive logic but we must add to these a requirement to adequately represent the operational aspects of the logic, \textit{q.v.} § 4.

3.1 Semantics of proofs

The semantics of intuitionistic logic and, to varying degrees, of substructural, classical and modal systems too, can be elegantly summarized by the so-called propositions-as-types-as-objects triangle, \textit{q.v.} Figure 1, in which (natural deduction) proofs correspond to (typed) $\lambda$-terms which correspond to classes arrows in categories with specified structure.
The triangle provides a framework within which the proof theory, model theory and computational interpretation of systems can be formulated. For example:

- Intuitionistic logic: Here the correspondence between propositions and types \([12,1]\) is particularly strong. At the propositional level, one obtains a close proof-theoretic analysis via the simply-typed \(\lambda\)-calculus which extend to the predicate level via dependently-typed \(\lambda\)-calculi. Semantically, Kripke’s possible-worlds models of propositional consequence can be generalized to interpret proofs. At the propositional level, bi-cartesian closed categories,\(^3\) such as presheaf categories of the form \(\text{Set}^W\), where \(W\) is a small category of worlds, are sufficient. At the predicate level, the essential structure of proofs is captured by fibrations, or indexed categories, with specified extra structure \([17,25,6]\), q.v. Figure 2:

  - The arrows \(\sigma\) in the base category are substitutions, i.e., maps between the sets \(Y\) and \(X\) of variables;
  - The arrows \(\Gamma \xrightarrow{\phi} \phi\) in the fibre \(F(X)\) over the set \(X\) of variables interpret proofs of sequents formed using the variables in \(X\);
  - Substitution lifts from the base to the fibres via the contravariance in the fibration:

\[
Y \xrightarrow{\sigma} X \quad \leftrightarrow \quad F(X) \xrightarrow{\sigma^*} F(Y);
\]

- **MILL** and **BI**: Here again the correspondence works well at the propositional level (no predicates or quantifiers). For **MILL** without exponentials, and so without any treatment of intuitionistic logic, we get a corresponding simply-typed \(\lambda\)-calculus which can be interpreted in a symmetric monoidal closed category \([20]\). For **MILL** with the exponential, \(!\), we once again get a simply-typed \(\lambda\)-calculus but it must be interpreted in a complex structure involving a monoidal adjunction between a symmetric monoidal closed category and cartesian closed category. For **BI**, we get a simply-typed \(\lambda\)-calculus, with both multiplicative (or linear) and additive (or intuitionistic) function spaces, which can be interpreted in a (bi-cartesian doubly closed category, i.e., a category which enjoys monoidal two closed structures, one

---

\(^3\) A CCC is bi-cartesian closed if it is also bi-cartesian, i.e., has co-products as well as products.
of which is bi-cartesian. At the predicate level, the analysis is less clear. The \( \lambda \Lambda \)-calculus \([14,13,15,16]\) is a dependently-typed \( \lambda \) calculus which provides a partial analysis, being both in the spirit of \( \text{BI} \) and yet somewhat reliant on the presence of a form of Dereliction. Nevertheless, \( \lambda \Lambda \) can be interpreted in the general fibred framework sketched in Figure 2;

- Classical logic: the \( \lambda \mu \)- and \( \lambda \mu \nu \)-calculi \([22,26,27,29,30]\) allow the presentation of a semantics of the proofs of propositional classical logic within the fibred framework \([29]\). In \( \lambda \mu (\nu) \), sequents are structured so as to have a chosen active position on the right (the left-most position),

\[
\Gamma \vdash t : \phi, \Delta,
\]

where the term \( t \) inhabits (in the sense of the propositions-as-types correspondence) the formula, \( \phi \), in the context of the formulæ in \( \Delta \). A fibred semantics is obtained by taking a base category in which the objects are (interpretations of) the \( \Delta \)s, the non-active formulæ, and the arrows are (interpretations of) structural operations, such as Weakening, Contraction and, critically, Exchange. The fibres are then cartesian closed categories in which the “active” sequent \( \Gamma \vdash t : \phi \) can be interpreted, subject to various structural conditions.

All of these examples, and many more besides, can be seen as fitting into
a model-theoretic framework based on a formulation of Kripke-Beth-Joyal semantics [19] in a general setting based on fibred (or indexed) categories [25,6,15,16]. The basic idea is that a model,

\[ M = \langle \text{Struct}, [-] \rangle \]

consists of structure, \( \text{Struct} \), together with an interpretation, \([-] \), of the syntax of the logic in the structure. The structure is given by a functor,

\[ \text{Struct} : [\mathcal{W}, [\mathcal{B}^{\text{op}}, \mathcal{V}]], \]

in which

- \( \mathcal{W} \) is a (small) category of worlds,
- \( \mathcal{B} \) is a (small) category which interprets the term language of the logic, and
- \( \mathcal{V} \) is a category of values which interprets the consequences, the sequents, of the logic.

We require the structure to carry additional structure, according to the logic to be interpreted. For example, to interpret intuitionistic predicate logic, we require, \textit{inter alia}, that \( \mathcal{B} \) have products, each fibre (object of \( \mathcal{V} \)) be cartesian closed and that certain functors between fibres interpret the quantifiers. To interpret the dependent type theory corresponding to intuitionistic predicate logic, we must impose a more intimate relationship between the base and the fibres by requiring that the product, or extension combinator, in the base be constrained to apply to the objects definable in the corresponding fibre [25,6].

Turning briefly to equality, the appropriate equivalence relation on proofs is derived from properties such as normalization and cut-elimination. Semantically, these properties correspond to properties such as coherence.

3.2 Semantics of proof-searches

We have seen that the propositions-as-types-as-objects correspondence provides a framework within which the semantics and proof theory of a wide variety of deductive logics can be formulated and analysed.

So it seems valuable to ask whether such a framework can be provided for the formulation and analysis of reductive logics. The desired set-up is sketched in Figure 3, in which \( \Gamma \vdash \phi \) denotes a sequent which is a putative conclusion and \( \Phi \Rightarrow \Gamma \vdash \phi \) denotes that \( \Phi \) is a search with root \( \Gamma \vdash \phi \). The judgement \( [\Gamma] \vdash [\Phi] : [\phi] \) indicates that \([\Phi]\) is a realizer of \([\phi]\) with respect to assumptions \([\Gamma]\). The definition of such a realizer is problematic and discussed below.

It is clear that provision of such a framework is a non-trivial problem. The main difficulty is that the objects constructed during a search, \textit{i.e.}, a reduction, are, in contrast to the objects, \textit{i.e.}, proofs, constructed during deduction,
inherently *partial*. Whilst any deduction proceeds from axioms to a guaranteed conclusion and so constructs a proof, searches proceed from a putative conclusion to sufficient premisses. At any intermediate stage, it can be that it is impossible to complete the search so as to obtain a proof, i.e., all possible reductions lead to trees in which there are leaves of the form $\phi \vdash \psi$ in which the formulae $\phi$ and $\psi$ are both distinct and irreducible.\(^4\)

Suppose, then, that we have a deductive system $D$ which is interpreted in a category $\mathcal{C}$. Consider the interpretation of an axiom sequent, $\phi \vdash \phi$, given by

$$[\phi] \xrightarrow{[\phi]} [\phi],$$

the identity arrow from $[\phi]$ to itself. Proof trees over $D$ have the property that all leaves have this form (or something very like it).

Now consider the reductive system $S(D)$, obtained by reading each of $D$’s inference rules as reduction operators. Search trees over $S(D)$ can have leaves of the form $\phi \vdash \psi$, where $\phi$ and $\psi$ are distinct, irreducible formulae, so that there is no way to reduce the leaf to an axiom of the deductive system. A semantics of searches in $S(D)$ must interpret leaves of this form.

One solution is to interpret searches not in the category $\mathcal{C}$ but in the polynomial category $\mathcal{C}[\alpha]$ over an indeterminate $\alpha$.\(^5\)

\[ \text{Aside: If } A \text{ and } B \text{ are objects of a category } \mathcal{C}, \text{ we can adjoin an indeterminate } A \xrightarrow{\alpha} B \text{ by forming the polynomial category } \mathcal{C}[\alpha]. \text{ The objects of } \mathcal{C}[\alpha] \text{ are the objects of } \mathcal{C} \text{ and the arrows of } \mathcal{C}[\alpha] \text{ are formed freely from the arrows of } \mathcal{C} \text{ together with the new arrow } \alpha. \text{ The basic ideas may be found in [19].} \]

Then the interpretation of a leaf of the form $\phi \vdash \psi$ can be defined as follows:

$$[\phi] \xrightarrow{\alpha} [\psi].$$

The corresponding language of realizers is the internal language of $\mathcal{C}[\alpha]$.

---

\(^4\) We say that an occurrence of a formula $\phi$, in a search tree over a system $S$, is *irreducible* if it is not the principal formula of an instance of any reduction operator of $S$.

\(^5\) In general, the polynomial over a set of indeterminates.
Whilst polynomials over categories of proofs provide a place within which
searches can be interpreted, there is much more to consider in the semantics of
search. The key point is that the denotational semantics of the search process
is inherently intuitionistic: the search procedure, let’s think of it as an agent,
can be seen as van Dalen’s creative subject [3] exploring a Kripke frame \((W, \sqsubseteq)\)
in which the ordering is generated by the reduction operators of the logic. At
each reduction, starting from a given endsequent, the creative subject increases
his knowledge of established atomic propositions. For example, illustrated in
Figure 4, suppose we have, in intuitionistic logic, the endsequent

\[
\phi; \phi \supset \psi; \phi \supset \chi \vdash (\psi \lor \psi') \land (\chi \lor \chi'),
\]

in which \(\phi, \psi, \psi', \chi\) and \(\chi'\) are atomic. At the root world, \(w_1\), the only atomic
proposition established on the left, potentially capable, in the presence of a
matching \(\phi\) on the right, of forming an axiom sequent \(\phi \vdash \phi\), is \(\phi\). \(^7\)

The next two reductions, \(\land R\) and \(\lor R\), take us to worlds \(w_2\) and then \(w_3\)
and \(w_4\) without adding to the atomic propositions established on the left. Next
comes a \(\supset L\), with principal formula \(\phi \supset \psi\). This step adds \(\psi\) to the atomic
formulæ established on the left, and so capable of contributing to axioms.
As before, the accession to worlds \(w_6\) and \(w_7\), via an \(\lor R\), adds no atoms to
the left. Finally, the \(\supset L\) leading to \(w_8\) adds \(\chi\) to the collection of formulæ
established on the left.

Each reduction increases (or, at least, does not decrease) the number of
established (atomic) propositions on the left. Of course, there is a potential
problem here with Weakening,

\[
\uparrow \quad \Gamma \vdash \psi \\
\quad \Gamma, \phi \vdash \psi \quad \text{W.}
\]

Such a reduction may lead to a world able to establish fewer facts than in its
ancestor, contradicting the usual hereditary condition in Kripke-like semantics.
However, we can work with presentations of intuitionistic and classical logic
in which Weakening is incorporated into the form of the axiom and so need
not be taken as a reduction operator.

Note that this point of view also makes sense substructurally. Consider,
for example, the endsequent

\[
\phi, \phi \rightarrow \psi \vdash \psi \lor \psi'
\]

in multiplicative intuitionistic linear logic \([9]\) (a similar example can be for-
mulated in BI \([21,23,24]\)). A sketch of a Kripke model generated by a search

\(^6\) We use “;” as the antecedent-constructor for intuitionistic logic to distinguish it from the
“,” used in a later example based on linear logic.

\(^7\) Our use of just atoms to form axioms should be considered analogous to the use of atoms
in a least Herbrand model \([5]\).
Fig. 4. Search Semantics as Kripke Semantics

for a proof of it is given in Figure 5. As in the previous example, the ∨R does not add any established atomic propositions to the left-hand side. The ¬⊙L is, however, more interesting. It adds, in this case ψ, to the collection of atoms which occur on the left on some branch of the search, which has two branches at this point, one being φ ?- φ and the other being ψ ?- ψ: the collection (multiset) of atoms occurring on some branch is thus {φ, ψ}. Of course, in order to ensure that such models do not violate various substructural conditions, such as a requirement that axioms be exactly of the form φ ?- φ, we must impose additional conditions. One way to do this might be via a forcing relation for the substructural connectives. If the worlds for such a semantics are drawn from a commutative monoid, then the world w_4 might be considered to decompose as a monoidal product of worlds, {φ} · {ψ}, where · is multiset union, with the component worlds {φ} and {ψ} forcing φ and ψ, respectively. Another way to do this, algebraically, might be along the lines of the method of Boolean constraint equations explained in [10].

The φ and ψ, here both established at w_4, occur in different sequents just as before. The difference here is that, in this system, there is no possibility of generating ⊑ using the operators corresponding to Contraction or Weakening,

\[ \Gamma, \phi \vdash \psi \quad C \quad \Gamma \vdash \phi \vdash \psi \quad W. \]

\[8\] If the monoid is also preordered, then it might also provide the worlds for a forcing semantics of the intuitionistic connectives (see [24,21,23]).
It follows that we might expect to interpret reductive logic in models based on families of polynomial categories indexed by Kripke worlds [27]. Note that possible-worlds aspect of this choice of semantics is somewhat independent of the underlying deductive logic: We have indicated that both intuitionistic and substructural systems can be treated in this way but so too can classical systems.

Turning briefly once again to equality, we must note an important difference between deduction and reduction, motivated by the very high computational complexity of proof-search problems. Consider, for example, following [35], searches for proofs of the sequent

\[ \exists x. \forall y. p(x, y) \rightarrow \forall v. \exists u. p(u, v). \]

One search is given in (1). This search fails because there is no way to instantiate the existential so as to form a deductive axiom. Note, however, that we reach a perfectly good leaf of the search which can be interpreted in the appropriate polynomial construction.

\[
\frac{?}{p(a, b) \rightarrow \exists u. p(u, c)} \ \exists L \\
\frac{p(a, b) \rightarrow \forall v. \exists u. p(u, v)}{\forall R} \\
\frac{\forall y. p(a, y) \rightarrow \forall v. \exists u. p(u, v)}{\forall L} \\
\frac{\exists x. \forall y. p(x, y) \rightarrow \forall v. \exists u. p(u, v)}{\exists L}
\]

(1)

However, all is not lost. A simple reordering, or permutation, of the reductions of the quantifiers yields the following search tree:
\[
\begin{align*}
p(a, b) & \rightarrow p(a, b) \\
p(a, b) & \rightarrow \exists u. p(u, b) \quad \forall R \\
\forall y. p(a, y) & \rightarrow \exists u. p(u, b) \quad \forall L \\
\forall y. p(a, y) & \rightarrow \forall v. \exists u. p(u, v) \quad \forall R \\
\exists x. \forall y. p(x, y) & \rightarrow \forall v. \exists u. p(u, v) \quad \exists L
\end{align*}
\]

As shown in [35,28], the existence of the latter search can be calculated from the former via the technology of reduction orderings. Such an ordering, which compares the structure of formulæ and substitutions with the structure of the search tree, induces an equality on searches which should be respected in the semantics.

Whilst it is clearly desirable to have a good semantics for searches, we are, of course, interested in calculating proofs in the underlying deductive logic, paying due attention to issues such as permutation, as discussed above. Our position in these notes, then, is that proof-search, \textit{i.e.}, reductive logic, demands an essentially intuitionistic semantics, even if the underlying deductive logic is classical, and that we must understand the relationship between the intuitionistic semantics of search and the semantics of the underlying deductive logic.

4 Control and its semantics

Deductive inferences are many-to-one: before an instance of an inference,

\[
\Downarrow \quad \text{Premiss}_1 \ldots \text{Premiss}_m \quad \text{Conclusion} \quad R,
\]

may be performed, each of the \( m \) premisses must be established and the order in which they are established is of little importance.\(^9\)

Reductive inference, in contrast, is one-to-many: in order to apply an instance of reduction,

\[
\Uparrow \quad \text{SufficientPremiss}_1 \ldots \text{SufficientPremiss}_m \quad \text{PutativeConclusion} \quad O_R,
\]

it is necessary only to have reduced as far the putative conclusion, yielding \( m \) sufficient premisses, \textit{i.e.}, \( m \) new putative conclusions.

It should now be clear that the pattern of choices of premisses made by the search procedure has a critical impact on the outcome of the search. For example, in systems such as Prolog [5,8,33], one premiss might lead to looping

\(^9\) More important, perhaps, in systems which combine top-down and bottom-up reasoning [11].
whereas another premiss might fail very quickly. These observations lead us to the view that

\[
\text{Reductive Logic} = \text{Reduction} + \text{Control},
\]

\textit{i.e.}, the specification of a reductive logic requires not only the definition of the reduction operators, corresponding to the rules of inference of a deductive logic, but also a definition of the strategy which controls their use. See [8] for a related discussion.

How, then, are we to formulate a semantics within which not only the logical aspects of search but also the operational aspects can be adequately represented? Well, as we observed in § 2, perhaps the key distinction between proof and search is the rôle of failure.

Consider a search, with the underlying deductive system being intuitionistic propositional logic, beginning with \(\phi, \phi \supset \psi \dashv \psi \lor \psi'\). Suppose we choose first to apply the operator corresponding to \(\lor R\), picking the branch on which we have \(\psi'\):

\[
\uparrow \quad \frac{\phi, \phi \supset \psi \dashv \psi'}{\phi, \phi \supset \psi \lor \psi} \quad \lor R.
\]

At this point, only remaining possible reduction is a \(\supset L\) with principal formula \(\phi \supset \psi\):

\[
\uparrow \quad \frac{\phi \dashv \phi \supset \psi \dashv \psi'}{\phi, \phi \supset \psi \supset \psi'} \quad \supset L.
\]

Now we can only fail because there is not way to complete the right-hand branch. Of course, we want to \textit{backtrack} to the point at which we chose the branch of the search space with \(\psi'\) and choose the branch with \(\psi\) instead. Backtracking, however, is computationally expensive and structurally non-too elegant.

There is, however, an appealing solution. We can formulate our presentation of intuitionistic logic so that the possibility of choosing \(\psi\) rather \(\psi'\) is recorded even on the branch of the search space corresponding to the choice of \(\psi'\). The key step is to use a multiple-conclusioned sequent calculus and reformulate the \(\lor R\) rule, and its corresponding reduction operator, as follows:

\[
\Gamma \vdash \phi, \psi, \Delta \quad \Rightarrow \quad \Gamma \vdash \phi \lor \psi, \Delta.
\]

Then the same sequence of reductions as the previous search now reaches

\[
\uparrow \quad \frac{\phi \dashv \phi \supset \psi \dashv \psi'}{\phi, \phi \supset \psi \lor \psi'} \quad \lor R.
\]

and the right-hand leaf can be closed (Weakening on the right can be handled just as Weakening on the left).
One way to understand this point of view is provided by the presentation of classical logic as the $\lambda \mu \nu$-calculus, a term calculus for the classical sequents which is analysed proof-theoretically in [29,30] and semantically in [26] (recall the discussion in §3). Recall that in this view, in which sequents are structured so as to have a chosen active position on the right (the left-most position),

$$\Gamma \vdash t : \phi, \Delta,$$

where the term $t$ inhabits (in the sense of the propositions-as-types correspondence) the formula, $\phi$, in the context of the formulæ in $\Delta$. The non-active part of the right-hand side, $\Delta$, is seen as a “scratchpad”, or storage zone. If we consider the intuitionistic search space as being embedded in the classical search space, as we can see our approach above, then we can use the scratchpad to record the choices made during search and the inhabiting term $t$ to provide access to the store. In our example, the requisite $\psi$ is stored in the sequent and can be accessed by exchanging it into the active position.

Semantically, the choice of embedding in the classical calculus is very helpful. We have seen that the semantics of $\lambda \mu \nu$ fits well within the fibred, “propositions-as-types-as-objects”, framework. However, a semantics of $\lambda \mu \nu$ based on a category of continuations can also be given and this semantics can be presented within the fibred framework [26]. Within this setting, backtracking is modelled by exchanging the active formula, which is interpreted as an arrow in the base category of the fibration.

There is, however, much more to control than backtracking. For example, in order to determine an execution strategy, we must choose

- which premiss to work on next,
- which instance of which rule to apply next, and
- which point at which a choice was made to which to backtrack.

Consider, for example, the execution of a presentation of a version of Prolog, based on the uniform proof procedure for hereditary Harrop formulæ, i.e., the class of formulæ mutually inductively defined by

\[
\begin{align*}
\text{Definite formulæ} & \quad D & := & \quad A \mid D \land D \mid G \supset A \\
\text{Goal formulæ} & \quad G & := & \quad A \mid G \land G \mid G \lor G \mid D \supset G.
\end{align*}
\]

Uniform proofs, viewed as reductions, are those in which right rules are always preferred over left rules where both are applicable. How can this structure be

\[\text{It is not clear that the move all the way to a classical calculus is necessary: a multiple-conclusioned presentation of intuitionistic logic [4] may be sufficient (cf. [29,30]).}\]

\[\text{Uniform proofs are complete for hereditary Harrop sequents, i.e., sequents which have exactly } D\text{-formulæ on the left and } G\text{-formulæ on the right. In fact, without loss of complete-}\]
characterized declaratively? The binary rules yield two premisses; how are we to specify declaratively the procedure to choose which premiss to reduce first? At a stage at which all possible right reductions have been performed, how are we to specify declaratively the procedure to choose which of the applicable instances of $\supset L$, e.g., the left-most, to apply?

One solution is to impose the following orderings:

• On the set of rule schemata:

$$\{(\land L) \sqsubseteq (\land R), (\lor R), (\supset R) \sqsubseteq (\supset L)\}$$

Here we suppress the specification of the ordering between the right rules. In single-conclusioned systems, no choice is possible; in multiple-conclusioned systems, it can depend upon the permutability properties of the calculus.

• For each binary rule, an ordering on the premisses:

$$\Gamma ?- G_1 \land G_2, \Delta \sqsubseteq \Gamma ?- G_1, \Delta \land R$$

To see how this works, consider the construction of a uniform proof of

$$D ?- A \land B, A \supset B,$$

where $A$ and $B$ are atomic. The ordering on rules directs us to work first on the right. Reducing the $\land$ takes us to

$$(1) \quad D ?- A, A \supset B \quad \text{and} \quad (2) \quad D ?- B, A \supset B.$$  

Now, the ordering on the premisses of the $\land R$ rule directs us to reduce (1) next, leading to

$$D, A ?- A, B$$

which is an axiom. At this point a success continuation directs us to the next (left-most, say) uncompleted branch; we soon succeed. Here we have suppressed the details of the exchanges required to move a chosen formula to the active position.

Of course, we should like to integrate ordering constraints such as these (we have greatly simplified the ideas for these notes) into the definition of a model of proof-search. To this end, we conjecture that working not merely with the interpretations of the syntactic constructs of the language but rather with algebraic actions, i.e., actions of orders, on these constructions. Thus completeness, we restrict the $\supset L$ rule to those cases in which the head of the implicational formula matches with the (necessarily) atomic goal. Also, we can initially perform all of the $\land L$ rules below any given $\supset R$, thereby reducing the applicable left rules to $\supset L$.

\[12\] Recall that we have restricted $\supset L$ to be unary.
we seek to import the technology of domain theory, which provides a well-developed account of order and limit, into the model-theoretic semantics of proof-search.

5 An application: the denotational semantics of logic programs

The concerns we have discussed are all very intriguing and amusing but are there any applications? Perhaps the most obvious application is to the denotational semantics of logic programs.

There are two main approaches to the semantics of logic programs.

• First, the $T_P$ operator used in the semantics of (intuitionistic) logic programs [5]. Here, a mapping, $T_P$, is defined from interpretations to interpretations, in which the image is the result of applying the rules of the program, $P$, to the initial interpretation via a combination of modus ponens and unification. The semantics of the program is then given by the least fixed point of this operator, a co-limit of ordinal powers of $T_P$. It is interesting to note that this forward chaining system is traditionally used to provide a fixpoint semantics for SLD-resolution [5], a backward chaining system (cf. the discussion in [11]). The $T_P$ semantics is very good logically. It gives a term model of the underlying logic and includes an explicit measure of the “logical complexity” of a goal, i.e., the power of $T_P$ at which the goal can be proved.

However, the $T_P$ semantics is quite poor in its treatment of the execution dynamics of $P$. For example, a given program $P$ can either terminate or loop depending upon the choice of strategy, yet the $T_P$ semantics will not distinguish the two systems. Our position is that different strategies determine different logics and that a good semantics must make such distinctions.

• Second, a denotational semantics of the execution of Prolog programs can be given in the usual domain-theoretic style [32].

Our position is that a good semantics of logic programs must account for both the logical and operational aspects of the meaning of a program.

6 Conclusion

We have introduced the problem of providing a semantics for reductive proof theory, i.e., for proof-search. We have sketched a solution which can be understood to develop in three steps:

• First, a basic, logically adequate semantics of searches, analogous to a semantics of proofs;
• Second, an extension of the basic semantics to handle backtracking; and
• Third, a further extension to handle strategies, thereby completing the mod-
elling of control.

Applications of this technology, other than to logic programming, which we discussed in § 5, might be to

- logical frameworks and their associated theorem provers, and
- computer algebra systems.

References


