Notre Dame Journal of Formal Logic
Volume 30, Number 2, Spring 1989

Modularity and Relevant Logic

JAMES GARSON

Abstract A practical system of reasoning must be both correct and efficient. An efficient system which contains a large body of information can not search for the proof of a conclusion from all information available. Efficiency requires that deduction of the conclusion be carried out in a modular way using only a relatively small and quickly identified subset of the total information. One might assume that data modularity is incompatible with correctness, where a system is correct for a logic L iff it proves exactly what is valid in L. We point out that modularity and correctness are indeed incompatible if the logic in question is classical. On the other hand, the two desiderata are compatible for relevance logic. Furthermore, Horn clause resolution theorem proving is modular (this helps explain its relative efficiency) and the logic for which it is correct is relevance logic not classical logic.

1 Introduction *The Modularity of Mind* [4] rallies the troops for a movement in cognitive science which has been gradually gaining strength over the past decade or two. The old view, championed, for example, by Newell and Simon's work [9] on the General Problem Solver, was that intelligence is *monolithic*; it is describable in a relatively simple way, and it has access to all the available data. The new wisdom has it that cognition can be best explained by assuming the existence of *modules* which have specialized functions, and narrow lines of communication with the data and their peers.

The strongest kind of evidence for rejecting monolithic theories of the human mind has come from work on perceptual processing, for example Marr's work [7] on vision. We have learned that some things which we take for granted, for example the ability to match images in our left and right eyes during binocular vision, are not at all easily explained. Given the relatively slow neural response times, and the speed of the overall system, we feel compelled to postulate the existence of an array of parallel structures each of which processes information in only a small portion of the visual field. Given this assumption and phys-

Received January 27, 1987; revised July 9, 1987

iological evidence, fairly concrete things can be be said about the algorithms that carry out the process of perception.

Evidence on vision can hardly provide a conclusive argument against the monolithic view of the mind. The relatively low-level processing that goes on during sensory input does not begin to exhaust what a mind is capable of doing. High-level processing such as solving problems and drawing conclusions would seem to be more paradigmatic functions of the mind. Fodor describes his modules as part of the "input system", underscoring their role as part of sensory processing, but he goes on to argue for the existence of a "central system" capable of integrating information across sensory domains. So a monolithic theory of a higher-level process such as *reasoning* is at the very least compatible with what Fodor says.[1]

In this paper, we examine a *modular* account of reasoning. A reasoner is perhaps best treated as an array of modules with special functions and limited lines of communication. Although the system as a whole might have access to all information, each module would have access to only a small portion of the available data. Because each module carries out a limited task, modular systems can be significantly faster than monolithic systems.

We do not intend to argue the merits of this account either as a description of the human mind or a blueprint for computer reasoning systems. The purpose of this paper is to examine the view that formal logics, and especially the classical predicate calculus, can provide an adequate account of efficient reasoning systems. We will conclude that classical logic and modularity are incompatible, but we will not conclude that logic has no place in the theoretical description of fast reasoning systems. Although predicate logic forces us to adopt the monolithic view of reasoning, at least one nonclassical logic, relevance logic, is compatible with the modular hypothesis. If relevance logic is used as the standard of accuracy for modular reasoning systems, it may be possible to escape the tradeoff between logical accuracy and speed which bedevils classical logic.

We have further evidence of the importance of relevance logic as a standard for efficient reasoning systems. Logic programming is a widely used and relatively efficient methodology in artifical intelligence research. It is based on a reasoning method called *procedural derivation* which we will show is compatible with modularity. We will prove that procedural derivation obeys the standards of relevance logic. So relevance logic has already been adopted (covertly) as a standard for efficient reasoning systems.

2 Acceptance of logic as a theory of reasoning

The earliest work on the application of formal logic to the construction of computer reasoning systems accepted the monolithic viewpoint. Logical reasoning was taken to be one thing: the verification of the validity of arguments in first-order logic. The main research problem was to find correct and efficient techniques for checking validity. The discovery of resolution theorem proving in the early sixties was a milestone in these efforts, because of the method's simplicity and relative efficiency (see [10]).

The thesis that predicate logic can (or even should) provide an adequate theory of reasoning has had rough going lately, particularly among artificial intelligence researchers interested in problem solving and natural language processing. The enthusiasm for resolution theorem proving faded in the late sixties as the costs of applying the method to more complex problems became more apparent. The problems with predicate logic are well known. First, there is no decision procedure for predicate logic, so no computer program can correctly answer all our questions about inference anyway. But what makes matters worse is that even propositional logic is NP-complete: it falls into a class of problems that are practically intractable because the time to compute an answer rises exponentially with the size of the problem.

The response of many computer scientists has been to abandon the approach based on logic, and to hand code specific inferencing behavior in their reasoning systems. "Semantic nets", "frames", "scenes", "servants", "demons", and "actors" are buzz words for techniques that help organize the programming effort that ensures that a given system reasons in the desired way. The trouble with this response, as everyone has been quick to admit, is that these techniques put no restrictions on what behavior can be programmed, so the resulting systems are not known to obey a *theory* of inference. Formal logic may be too expensive in computer time, but at least we have consistency and completeness results which assure us that conclusions are correctly drawn from the data.

Disenchantment with formal logic has now spread to philosophers. Cherniak [2] argues that the computational complexity of predicate logic challenges not only its usefulness but also its universal acceptability. The conclusion of his argument is that a reasoning computer or human may not rely on classical predicate logic if it or he wants to get anything done. At best, it must rely on "clever hacks" which occasionally make errors, but which compensate for this by providing timely answers. Since predicate logic is impractical, it cannot be accepted as a realistic standard for reasoning.

We agree with Cherniak that any sensible theory of reasoning must make clear how timely computation of answers to common sense questions can be calculated. However, we are unwilling to give up on logic so quickly. There is territory yet to be explored between monolithic systems based on predicate logic on the one hand, and *ad hoc* programming on the other.

This paper explores a model where reasoning systems consist of an assembly of *logical* modules, logical in the sense that each one meets the conditions of adequacy (consistency and completeness) which have guided our construction of logics. The idea is that a reasoner is an array of small logical systems each of which is capable of answering questions both quickly *and* accurately in its own domain of expertise. The purpose of presenting this model is to show that (contrary to the conclusions one might draw from Cherniak) speed and logical accuracy need not be incompatible.

Cherniak considers modularity as a possible solution for the problem of efficiency, but he objects (p. 753) that modules would "be particularly weak on inferences . . . distributed between such (modules)". However, if modules are *logical*, as we will propose, no module will ever need to consult data outside its domain in order to reason correctly, and so there will *be* no inferences distributed between modules. Cherniak's assumption that there will be such inferences makes sense if one adopts classical logic. On the other hand, we will show that if relevance logic is adopted as the logical standard, then it is at least possible to construct a system of modules which is fast and completely accurate. It would be fast because each problem would be referred to a module which contains a small portion of the total data and resources. It would be accurate because the system as a whole is consistent and complete for relevance logic.

3 The definition of logical modules

A logical module meets the standards of accuracy for some system of logic. It comes to a conclusion in its domain of expertise just in case that conclusion is *entailed* by the data available to the entire system. There are differences of opinion about which logic serves best as the standard for defining entailment. One of the purposes of this paper will be to argue backwards from the thesis of modularity to the logics that can serve as standards of accuracy. So as not to prejudge the issue, let us simply refer to our standard system as *global logic*, or *G* for short.

We will assume that a *reasoner* consists of a set of *modules* and a *selector*. Each *module* consists of three items: a set of rules that define the set of sentences that comprises its *domain of expertise*, a set of *inference rules*, and a set of sentences that serve as its *data*. The *global data* for the reasoner consists of the data in all the modules, and the *global domain* contains all sentences in the domains of all modules. The *selector* assigns to each sentence in the global domain a module (whose domain contains that sentence) which determines whether or not that sentence follows from the data. If the reasoner is to be efficient, the selector must be able to pick a module for a given sentence quickly, for example by examining the logical notation or vocabulary that it contains.

A sentence *is correct* (for *G*) just in case it is provable in *G* from the global data. A sentence is *provable in a module* just in case there is a proof of that sentence from the module's data, using the module's inference rules. A module is *locally consistent* for the global logic *G* just in case any sentence provable in a module is correct, and a module is *locally complete* (for *G*) just in case every sentence in the module's domain that is correct is provable in the module. A module is *logical* (for *G*) just in case it is both locally consistent and locally complete. A sentence is *provable in a reasoner* just in case the sentence can be proven in the module that is selected for it.

It is interesting to note that the assumption that all the modules of a reasoner are logical entails that the reasoner prove exactly what is correct. When all the modules are logical, the desired performance of the whole system is the sum of the performance of its parts.

4 Examples of logical modules

Let us give some examples of logical modules.[2] Conjunction modules can be constructed for the domain of sentences containing & as the only logical connective using the two rules Conjunction In and Conjunction Out. These rules are incomplete for the full range of inferences of predicate logic, but they are complete and consistent for arguments expressed in this domain. Not only that, the system has a very fast decision procedure: an argument is valid just in case every atomic conjunct of the conclusion appears as a conjunct in the data.

A second useful technique for checking entailment is the semantic net. It works for atomic sentences which express facts about transitive binary relations. Labeled arcs are used to represent each atomic sentence in the data along the lines of a Haus diagram. For example, the sentence *aRb* is represented by the diagram:

$$\begin{tikzcd} a \arrow[r, bend left, "R"] & b \end{tikzcd}$$

To check whether or not the sentence *cRd* follows from the data using a semantic net amounts to seeing whether there is a path from *c* to *d* following arcs labeled by *R*.

The trouble with these modules is that they handle a very limited set of arguments. A stronger sort of module which is widely employed and well understood is the system based on resolution theorem proving. Although the method runs in exponential time, its performance is good enough at actual tasks to find many practical uses. There is a particularly efficient form of resolution theorem proving for Horn clauses which we call procedural derivation. (Horn clauses will be described in Section 7.) Procedural derivation is fast enough to serve as the foundation of the Prolog computer language. As we will see, this efficiency can be attributed partly to the fact that procedural derivation is correct with respect to relevance logic, but not classical logic.

Table look-up can also be used as the underlying algorithm for checking validity in many domains. If the domain of a module is restricted to a finite list of questions, the answers to which have previously been computed (by whatever means), then validity can be checked by inspecting lists. Using this technique, a reasoner can eliminate expensive processing time by 'compiling out' those classes of inference which are especially common.

5 Logical modules and Fodorian modules

We can help motivate our definition of logical modules by showing that they have many of the properties which Fodor ([4], part III) attributes to input modules, or at least properties analogous to them. First, our modules are *domain specific*, for each system is designed to draw conclusions over a limited domain of expertise. Second, the accuracy condition ensures that operation of our modules can be *mandatory*. Once a sentence falls into a module's domain, the module can correctly handle it, so it is all right if the selector assigns a sentence to any module which has that sentence in its domain. Third, a Fodorian module should have *limited access* to inputs of the system. In our case, a module contains all the data it needs to solve problems in its domain, so that the only input it needs is the question to be answered. Fourth, a module should be *fast*. There are two reasons that logical modules are likely to be fast. First, each module may use a validity detecting algorithm specially designed for its data and domain. Second, the collection of the data of a module is assumed to be significantly smaller than the global data, so that even if algorithms with exponential complexity are used, the size of the problem is small enough to ensure practical response times. Fifth, a module should be *informationally encapsulated.* This means that the module does not need to access global information concerning the problem to be solved. Logical modules qualify because they have on board all the information they need.[3]

There is another important property of logical modules which does not fall on Fodor's list. We want modules to be *independent* in the sense that the addition or deletion of other modules does not significantly affect their behavior. Logical modules are independent because the reasoner will prove the sentences which are selected for a given module so long as that module exists in the system. The presence or absence of other modules makes no difference to its behavior. Independence is important because it ensures the graceful degradation of the behavior of the reasoner when processing or memory limitations have the effect of disabling certain modules.

*6 **Contradiction and modularity*** We have now laid the groundwork for discussing the relationships between systems of logical modules and their global logics. We will now argue that classical logic cannot possibly serve as the theory of a practical modular system, using a variant of an argument that is familiar to advocates of relevance logic.[4] The problem with classical logic is that it accepts the Law of Contradiction, i.e., that a contradiction entails any sentence. This principle is fatal to modularity. In a practical system, the global data will be large and changing. Any attempt to ensure its consistency will be extremely costly. What makes matters worse, no *reliable* consistency check is even possible, because there is no decision procedure for predicate logic. A practical system, then, must tolerate contradictions in its global data. If there are a fair number of modules, it is very likely that a contradiction in the global data will not appear in the data of any one module, but will be the result of a disagreement between modules. If the Law of Contradiction is obeyed under these circumstances, then the set of theorems of the global logic is the set of all sentences of the language. The result is that all modules must prove all sentences in their domains if they are to be locally complete. But they cannot, even if each one obeys the Law of Contradiction, because none of them finds a contradiction in its own data.

Complaints about the Law of Contradiction are commonplace, particularly in the literature on artificial intelligence. The difficulty is one reason commonly given for abandoning logic, and turning to hand coding in the design of inferencing systems. The objection is usually that reasoners based on classical logic are impractical because they "degrade ungracefully" when contradictions arise. Our objection to classical logic is more theoretical. Classical logic is not only impractical, it is also a poor *theory* about the inference behavior of modular systems. If standards for logical adequacy of modular systems are to be found at all, we must find them in nonclassical logics.[5]

*7. **Relevance and local completeness*** If logic is to play a prescriptive role in the theory of modular reasoning systems, we must choose a logical standard which allows the modules to have access to only part of the global data. Luckily, we have the prospect of achieving this with relevance logic (see [1]). Relevance logic does not accept the Principle of Contradiction, and it provides us with the tools to partition the global data into smaller parcels without jeopardizing the local completeness of the corresponding modules.

One of the fundamental ideas in relevance logic is to keep track of which sentences are used in the derivation of other sentences. If a derivation of sentence C is possible given sentence A, it does not follow that A relevantly implies C, for A may not have played any role whatsoever in the proof of C. One of the signs that a formula has played a role in the proof of another is if they contain common notation. In many relevance logics, A implies C only if there is a variable common to both A and C. This property allows the data that are relevant for proving a given conclusion to be neatly circumscribed.

To explain how this is done, let us define the *topic* of a sentence C (for a given set of data D) as follows. Every subformula of C is in the topic, and if any sentence in the data D contains a sentence in the topic then its subformulas are also in the topic. It is possible to show that any proof of C involves only data in the topic of C in a fragment of relevance logic. (See the Appendix where this definition is sharpened considerably and where proofs are given.) Given a group of sentences with the same topic, we can build a module with that topic as data which is locally complete with respect to relevance logic.

The moral of this discussion is that the difficulties which lead us to abandon classical logic do not require us to abandon logic altogether. A nonclassical logic such as relevance logic offers hope that the behavior of modular reasoning systems can be rationalized. Relevance logic is a standard candidate for modular reasoning systems just because it allows a method for narrowing down the data and rules that can possibly play a role in the proof of a given conclusion. This property is essential if locally complete modules are to be defined.

8 *Topic and the problem of modularizing data* It is widely recognized that the fundamental difficulty in designing efficient reasoning systems is to find ways to locate quickly a small set of data which is relevant to answering a given question. So far, the problem has been intractable in general, and the recognition of this has motivated the *ad hoc* approach to the design of inferencing systems, where the desired inferencing behavior is merely programmed in. Clearly, the definition of 'topic' that we have proposed, however artfully improved upon, could not itself provide a general solution to the problem of modularizing information. It is highly unlikely that any criterion which looks only at the syntactic form of the data could provide an answer to such a fundamental question.[6]

The reader should not suppose that our definition of 'topic' is meant to provide necessary and sufficient conditions for modularizing data for inferencing systems. Our goal is much more modest. It is to lay out the formal conditions that must hold if formal logic is to serve as a theory of modular reasoning. That the data fall into the topic of a question is sufficient to ensure that the data form a logically correct module for that question in relevance logic, but it is certainly not a necessary condition. In all likelihood, further techniques for organizing data will be needed to ensure efficient behavior from a complex system. Further difficulties may be encountered in providing efficient algorithms for assigning data to given modules when data are added to the system. Although this process need not be as fast as question answering, it should not take so long that updating the system becomes prohibitively expensive. There is no guarantee (nor could there be one as far as I can see) that any one proposal for modularizing data could ensure both logically accurate and efficient behavior, even when the logical standard is relevance logic.

Our purpose in defining 'topic' has been to argue that while classical logic cannot serve as a standard of modular reasoning, relevance logic is at least not open to the same objection, and so it is at least *possible* for modular reasoning systems to meet the logical standards imposed by relevance logic. Whether an efficient system (say an expert system or a human organism) actually does (or even should) meet these standards is another matter to be argued in another place.

9 *Relevance logic and procedural derivation*

Relevance logic not only shows promise as a standard for modular reasoning systems, but it has, in a sense, been already adopted by artificial intelligence researchers. The resolution method for Horn clauses *appears* to be based on classical logic, but *procedural derivation* (see [6]), the method actually used for logic programming, is not complete for classical logic, and is in fact equivalent to relevance logic.

Horn clauses are sentences of the form:

$$A_1, A_2, \ldots, A_n \Rightarrow C$$

where the sentences $A_1, \ldots, A_n$ and C are atomic. A Horn clause of this form is read '$A_1, A_2, \ldots,$ and A_n entail C'. The list $A_1, A_2, \ldots, A_n$ of *antecedents* may be empty, in which case $\Rightarrow C$ asserts that C. The *consequent* C may also be missing in which case $A_1, A_2, \ldots, A_n \Rightarrow$ says that the antecedents entail a contradiction.

Although the reading usually given for the Horn clause $A_1, \ldots, A_n \Rightarrow C$ is the propositional logic expression 'if A_1 *and*...*and* A_n then C', this is the wrong reading if we are translating Horn clauses into the notation of relevance logic. If the symbol '$\Rightarrow$' represents relevant entailment, we may not represent $A_1, \ldots, A_n \Rightarrow C$ by $(A_1 \,\&\ldots\&\, A_n) \rightarrow C$. The reason is that $(A_1 \,\&\ldots\&\, A_n) \rightarrow C$ may be provable in relevance logic when the sentences $A_1, \ldots, A_n$ do not relevantly entail C. For example, $(p \,\&\, q) \rightarrow p$ is provable in R, but p is not derivable relevantly from the list p, q, since q is irrelevant to the proof of p. In order to interpret the clause $A_1, A_2 \Rightarrow C$ as a *relevant* implication, we use the formula $A_1 \rightarrow (A_2 \rightarrow C)$, which (by the deduction theorem) is provable just in case C relevantly follows from A_1 and A_2.

A *procedural derivation* of a sentence C begins by searching for clauses in the data whose consequents are C.[7] If a clause with consequent C is found that has no antecedents then the proof is complete, because this clause asserts C. If a clause with consequent C contains antecedents, then the method is applied again to try to prove each antecedent. If all antecedents of the clause are provable, then so is C. If no clause contains C as a consequent, then C is not provable.

Procedural derivation is not complete for classical logic: it misses some proofs that depend on the Law of Contradiction. To illustrate this, suppose the data contain only the clauses:

$$\Rightarrow A$$

and

$$A \Rightarrow.$$

The first asserts and the second denies A, and so by classical logic C follows. Imagine now that we are attempting a procedural derivation of C from these data alone, and that C is not A. Since no clause with C as a consequent is found in the data, C has no proof.

Procedural derivation has the advantage that the search for a proof is restricted to those clauses in the data which meet a condition of relevance. Just as in relevance logic, each conclusion has a topic which limits the data which could possibly enter into its proof. For procedural derivation, the *topic* of the sentence C is defined recursively to contain all clauses with C as a consequent, and all clauses that have a consequent which is also an antecedent of some member of the topic. (Actually, the definition of the topic is somewhat more complex in the case of quantified arguments. The details are given in the Appendix.) A proof of a sentence using procedural derivation involves only members of its topic. If the data for a module are restricted to the joint topic of the sentences in its domain of expertise, then it will contain all the data it will ever need to complete a procedural derivation.

Of course, we still need to prove that procedural derivation is more than a "clever hack". We have no assurance that it corresponds to any logical system for which we have a suitable adequacy proof. Luckily, however, we can show that a module that uses procedural derivation as its inference rule, and whose data contain all sentences in the topic of each sentence of its domain, is locally consistent and complete for the concept of entailment generated by relevance logic. As we will show in the Appendix (Theorem 2), an argument with premises (data) D and conclusion C has a procedural derivation just in case the result of translating D (using $\rightarrow$, not &) into relevance logic relevantly entails the translation of C. It follows then that systems of modules which are actually used in computer science have the feature that relevance, not classical, logic provides a theory of their behavior.

In the course of proving the correctness of procedural derivation with respect to relevance logic, we also prove a result (Theorem 1) that sheds new light on the relationships between relevance and classical logic. Resolution theorem proving is significantly more powerful than procedural derivation, and we know that it is correct with respect to classical logic. We present a simple modification of resolution theorem proving that is compatible with modularity and which is correct with respect to relevance logic. The idea behind the variation is extremely simple. For a *relevant* resolution of C from D there must be a derivation of C from D by resolution in which all the members of D are *used.* So the adoption of relevance logic is painless; systems based on relevance logic can be easily implemented and fine tuned using the large body of knowledge we have already developed on theorem proving.

10 Conclusion The conditions that make a workable system of logical modules possible are not easily satisfied. We must abandon the Law of Contradiction, and turn to some nonclassical logic such as relevance logic for our logical standard. There are practical problems as well, for we have no guarantee that the global data will allow *any* definition of topic for the logic we have chosen that will result in an efficient system. There may be, after all, a necessary trade-off between logical accuracy and efficiency.

On the other hand, the results on procedural derivation and relevance logic show that at least some relatively efficient reasoning systems, based on techniques in wide use, do count as modular logical systems in the sense that the data relevant to answering a given question can be circumscribed in a reasonable way. Although decisions about how to structure the data in order to produce an efficient set of logical modules will be difficult ones to make, at least we have some hope that the desire for efficiency need not lead us to logical anarchy. If formal logic is to play its traditional role as a theoretical standard for reasoning systems, then a desire for efficiency requires us to carry on with the necessary research in nonclassical logic.

Appendix The purpose of this Appendix is to prove two claims made in the text. We show that procedural derivation is both consistent and complete with respect to the negation implication and quantifier fragment of RQ (see [8]), hereafter referred to as RQ.[8] We then use this result to prove that the derivation of a conclusion in that system never needs to appeal to data outside the topic of that conclusion. These results will allow us to improve the definition of the topic of a sentence.

The proof of the equivalence of procedural derivation and relevance logic depends on a stronger result (Theorem 1 below) concerning a relevant version of the rule of resolution. A simpler proof of the equivalence of procedural derivation and relevance logic is possible. However, Theorem 1 is interesting in its own right because it shows that relevance logic can serve as a standard for a variation of resolution which uses clausal form notation. So the result is not limited to the less expressive Horn clause notation used in procedural derivation.

We begin by explaining the notion of relevant resolution that we will show is equivalent to a relevance logic. The underlying idea is simple: clause C has a proof from the set of clauses D by *relevant resolution* ($D \vdash_{\mathrm{RR}} C$) iff there is a proof of C by repeated applications of the rule of resolution *which uses exactly the sentences in D.* We may define $D \vdash_{\mathrm{RR}} C$ by means of an axiom and rule as follows:

Axiom $C \vdash_{\mathrm{RR}} C$

Rule of Relevant Resolution
$$\frac{D \vdash_{\mathrm{RR}} L \Rightarrow A, R \qquad D' \vdash_{\mathrm{RR}} L', A' \Rightarrow R}{D, D' \vdash_{\mathrm{RR}} s(L, L' \Rightarrow R, R')}$$

where D and D' are sequences of clauses, C, C', and C'' are clauses, L, L', R, and R' are sequences of atoms, A and A' are atoms, and s is any substitution of terms for variables such that $s(A) = s(A')$. (We write $s(A)$ for the result of applying the substitution s to A.) Since the operations of permutation and *con-*

traction (removing duplicates) have no meaningful effect on any of the sequences we discuss in this Appendix, we treat sequences as sets whenever we like, though we always use sequence notation.

Resolution is not complete for classical logic. (For example, resolution cannot generate $A \Rightarrow A$ from the empty set of data.) However, the rule is complete for *refutations*, that is, if D is inconsistent in first-order logic, then there is a resolution proof of the empty clause $\Rightarrow$ from D. We will prove a similar result for relevant resolution.

We must first give a transformation from sentences in clausal form into corresponding sentences of relevance logic. Since function symbols and constants are available in clausal form, we assume their presence in the language of RQ. We begin with two notational matters. We drop parentheses associated with the conditional $\rightarrow$, and assume that they are replaced from right to left, so that $A \rightarrow B \rightarrow C \rightarrow D$ (for example) amounts to $(A \rightarrow (B \rightarrow (C \rightarrow D)))$. Second, open sentences of RQ are taken to be abbreviations of their universal closures.

The *corresponding sentence* for clause $A_1 \ldots A_n \Rightarrow C_1 \ldots C_m, C_{m+1}$ is $A_1 \rightarrow \ldots \rightarrow A_n \rightarrow {\sim}C_1 \rightarrow \ldots \rightarrow {\sim}C_m \rightarrow {\sim}{\sim}C_{m+1}$. If the consequent of the clause is empty, so that the clause has the form $A_1 \ldots A_n, A_{n+1} \Rightarrow$, then the corresponding sentence is $A_1 \rightarrow \ldots \rightarrow A_n \rightarrow {\sim}A_{n+1}$. Note that $\Rightarrow A$ corresponds to A, and $A \Rightarrow$ corresponds to ${\sim}A$. We define RQ-provability for hypotheses in the standard way: $D_1 \ldots D_n \vdash_{\overline{\text{RQ}}} C$ iff the sentence $D_1 \rightarrow \ldots \rightarrow D_n \rightarrow C$ is provable in RQ. A set $D_1 \ldots D_n, D_{n+1}$ is RQ-inconsistent (written $D \vdash_{\overline{\text{RQ}}} \Rightarrow$) iff $D_1 \ldots D_n \vdash_{\overline{\text{RQ}}} {\sim}D_{n+1}$. To simplify our discussion we will use clausal form notation as an *abbreviation* for the corresponding sentence of R. (For reasons we explained in Section 9, we are using *intensional* conjunction (and disjunction) when we translate clauses into R.)

Theorem 1 $D \vdash_{\overline{\text{RR}}} \Rightarrow$ *iff* $D \vdash_{\overline{\text{RQ}}} \Rightarrow$.

Proof: (left to right) The consistency of relevant resolution with respect to relevance logic may be proven by showing that relevant resolution is a derivable rule of RQ. The result depends on well-known features of RQ. First, we may freely permute and contract the consequent or the antecedent of $L_1 \ldots L_n \Rightarrow R_1 \ldots R_m$. (Remember we use clausal form notation to abbreviate corresponding sentences of RQ.) Notice that to contract a rightmost duplicate we need to appeal to "indirect proof": $({\sim}A \rightarrow A) \rightarrow A$, and to permute a rightmost pair we must appeal to contraposition: $({\sim}A \rightarrow C) \rightarrow ({\sim}C \rightarrow A)$. The second is a principle of generalized transitivity, namely that if $\vdash_{\overline{\text{RQ}}} L \Rightarrow A$ and $\vdash_{\overline{\text{RQ}}} A \rightarrow C$ then $\vdash_{\overline{\text{RQ}}}$ L $\Rightarrow$ C (see [1], p. 27). Given generalized transitivity and permutation, it is not difficult to show the derivability of the following rule which we call relevant cut:

$$\frac{D' \vdash_{\overline{\text{RQ}}} L' \Rightarrow R', A \qquad D \vdash_{\overline{\text{RQ}}} A, L \Rightarrow R}{D, D' \vdash_{\overline{\text{RQ}}} L, L' \Rightarrow R, R'}.$$

The axiom is clearly provable in RQ. To show that the rule of relevant resolution is derivable, suppose that $D \vdash_{\overline{\text{RQ}}} A, L \Rightarrow C$ and $D' \vdash_{\overline{\text{RQ}}} L' \Rightarrow R', A'$, and that $s(A) = s(A')$. Since the variables of $A, L \Rightarrow C$ and L', R', A are understood as universally bound, we obtain $D \vdash_{\overline{\text{RQ}}} s(A,L \Rightarrow R)$ and $D' \vdash_{\overline{\text{RQ}}} s(L' \Rightarrow R',A')$ by Universal Instantiation. Since $s(A) = s(A')$, we obtain $D,D' \vdash_{\overline{\text{RQ}}} s(L,L' \Rightarrow R,R')$ by relevant cut.

The proof from right to left amounts to showing that relevant resolution is complete with respect to relevance logic. Semantical methods traditionally used in the corresponding proof for classical logic are not helpful here because we must keep track of which sentences are *used* in a derivation in order to demonstrate the existence of a relevant resolution. Our strategy will be to make use of the Gentzen formulation GRQ of RQ. (GRQ is the result of adding the standard Gentzen rules to the system *GR* in [3].)

The axiom and rules of *GRQ* are as follows:

Axiom $A \vdash A$

Rules

$$(\vdash\rightarrow)\ \frac{D,A \vdash B,E}{D \vdash A \rightarrow B,E} \qquad (\rightarrow\vdash)\ \frac{D,B \vdash E \quad D' \vdash A,E'}{D,D',\ A \rightarrow B \vdash E,E'}$$

$$(\vdash\sim)\ \frac{D,A \vdash E}{D \vdash {\sim}A,E} \qquad (\sim\vdash)\ \frac{D \vdash A,E}{D,{\sim}A \vdash E}$$

$$(\vdash\forall)\ \frac{D \vdash At,E}{D \vdash \forall xAx,E} \qquad (\forall\vdash)\ \frac{D,At \vdash E}{D,\forall xAx \vdash E}$$

where the term t in $(\vdash\forall)$ must not appear in the conclusion of the rule. (Both D and E in these rules are sequences of sentences.) It is not difficult to show that whatever is provable in RQ is provable in GRQ. The only complication is showing that cut is admissible in GRQ, which is a straightforward exercise. It remains to be shown that if $D \vdash_{\overline{\text{GRQ}}} \Rightarrow$, then $D \vdash_{\overline{\text{RR}}} \Rightarrow$.

In order to show this we need to relate provable sequents $D \vdash E$ to corresponding refutations. Let us say that a sentence (of RQ) is *clausal* iff it corresponds to a sentence in clausal form. When A is clausal and corresponds to the clause $L_1 \ldots L_n \Rightarrow R_1 \ldots R_m$ the *opposite* of A (for sequent $D \vdash E$) is the sequence of clauses $s(\Rightarrow L_1), \ldots, s(\Rightarrow L_n), s(R_1 \Rightarrow), \ldots, s(R_m \Rightarrow)$, where s is a substitution of terms for variables of A such that all terms are new to D and E. In the case of the negation ${\sim}A$ of a clausal sentence the *opposite* is simply A.

Given the sequent $D \vdash E$, we form a corresponding refutation $D,E^* \vdash \Rightarrow$, where E^* is the result of taking the opposite of each member of E. The sequent $D \vdash E$ is *clausal* just in case all sentences of D and E^* are clausal.

Lemma 1 *If $D \vdash E$ is clausal, and $D \vdash_{\overline{\text{GRQ}}} E$, then $D,E^* \vdash_{\overline{\text{RR}}} \Rightarrow$.*

(Once Lemma 1 is proven we will have as a special case that if $D \vdash_{\overline{\text{GRQ}}} {\sim}C$ then $D,C \vdash_{\overline{\text{RR}}} \Rightarrow$. Since GRQ and RQ are equivalent, we have by the definition of $D \vdash_{\overline{\text{RQ}}} \Rightarrow$ that if $D \vdash_{\overline{\text{RQ}}} \Rightarrow$, then $D \vdash_{\overline{\text{RR}}} \Rightarrow$, which completes the proof from right to left.)

Proof: The proof of Lemma 1 is by induction on the structure of the proof of $D \vdash E$ in GRQ. We show that the axiom has a relevant refutation and that the rules preserve relevant refutation. Notice that all rules of GRQ have the property that if their conclusions are clausal, then so are their premises. It follows that any derivation of a clausal sequent in GRQ contains only clausal sequents. As a result, we may assume that A in the axiom corresponds to a clause $L_1 \ldots L_n \Rightarrow R_1 \ldots R_m$. We must show that we can derive $\Rightarrow$ from $L_1 \ldots L_n \Rightarrow R_1 \ldots R_m$ and $\Rightarrow s(L_1) \ldots \Rightarrow s(L_n)$, $s(R_1) \Rightarrow \ldots s(R_m) \Rightarrow$, and this is easily done with $n + m$ applications of resolution.

The cases of the rules for negation and ($\vdash \rightarrow$) are simple. We now consider ($\rightarrow \vdash$). Note that all quantifiers in a clausal sentence have widest scope, and no clausal sentence contains free variables. We know that every step of a derivation in GRQ is clausal: Any use of ($\rightarrow \vdash$) must produce a sequent $D, A \rightarrow B \vdash E$ where $A \rightarrow B$ has $\rightarrow$ as its main connective. It follows that no quantifiers appear in $A \rightarrow B$, and hence the clause to which it corresponds contains no variables. Since the sentence $A \rightarrow B$ is clausal, then it must have one of two forms: $A, L \Rightarrow R$ or $\Rightarrow A', R$, where $A = {\sim}A'$. In the first case we must show that

if (1) $D,\ L \Rightarrow R,\ C^* \vdash_{\overline{RR}} \Rightarrow$
and (2) $D',\ A \Rightarrow,\ C'^* \vdash_{\overline{RR}} \Rightarrow$,
then (3) $D,\ D',\ A, L \Rightarrow R,\ C^*,\ C'^* \vdash_{\overline{RR}} \Rightarrow$.

Given (1), we know there is a derivation tree with members of $D,\ L \Rightarrow R,\ C^*$ at the leaves and $\Rightarrow$ at the trunk. Add A to the antecedent of $L \Rightarrow R$ and all its descendents in the derivation tree. The result has an instance of resolution at each node. Since A contains no variables, the new derivation concludes with $A \Rightarrow$, and we know that $D,\ A, L \Rightarrow R,\ C^* \vdash_{\overline{RR}} \Rightarrow$ A. This together with (2) yields (3) $D,\ D',\ A, L \Rightarrow R,\ C^*,\ C'^* \vdash_{\overline{RR}} \Rightarrow$.

In the case where $A \rightarrow B$ corresponds to a clause of the form $\Rightarrow A', R$, where $A = {\sim}A'$, we must show that

if (1) $D,\ \Rightarrow R,\ C^* \vdash_{\overline{RR}} \Rightarrow$
and (2) $D',\ \Rightarrow A',\ C'^* \vdash_{\overline{RR}} \Rightarrow$,
then (3) $D,\ D',\ \Rightarrow A', R,\ C^*,\ C'^* \vdash_{\overline{RR}} \Rightarrow$.

The proof is symmetrical with the previous case, by adding A' to the consequent of $\Rightarrow R$ and its descendents in the derivation tree for (1).

We must now consider the axioms for the quantifier. Notice that when a sentence of the form $\forall x A x$ corresponds to a clause the clause itself has the form Ax. Therefore, in the case of ($\forall \vdash$) we must show:

if (1) $D,\ At,\ C^* \vdash_{\overline{RR}} \Rightarrow$
then (2) $D,\ Ax,\ C^* \vdash_{\overline{RR}} \Rightarrow$.

To do so take the derivation tree of (1) and replace x for t in At and its descendents on the tree. Note that replacing a variable for a term in an instance of resolution is a new instance of resolution, and so the new tree is a derivation of (2).

In the case of ($\vdash \forall$) we must show that

if (1) $D,\ At^*,\ C^* \vdash_{\overline{RR}} \Rightarrow$
then (2) $D,\ Ax^*,\ C^* \vdash_{\overline{RR}} \Rightarrow$

where t is foreign to the conclusion. Let t' be the constant chosen for x in forming Ax^*. Then Ax^* contains t' exactly where At^* contains t, and t' does not appear in D, and C^*. Replace t' for t in At^* and its descendents in the derivation tree for (1), noting that each node of the new tree is a new case of resolution. The result is a derivation of (2).

We have completed the proof of Theorem 1, and may now give the proof of the adequacy of procedural derivation with respect to relevance logic. Here we will be considering arguments that can be expressed by *Horn clauses*, that is, clauses with no more than one consequent.

We begin with definitions concerning procedural derivation. First, a *denial* is a clause $L\Rightarrow$ with an empty consequent. Next, a *procedural derivation* of the atom C from data D is a derivation by resolution of $\Rightarrow$ from D, $C\Rightarrow$ with the following features: The derivation begins by resolving $C\Rightarrow$ with some sentence in the data. Each subsequent step of the derivation involves a sentence from the data and the denial that results from the previous step. A set D of data is *used* in a procedural derivation just in case resolution was applied to each member of D in the course of the derivation. We write '$D \vdash_{\mathrm{PD}} C$' when C has a procedural derivation from D which uses all of D, $C\Rightarrow$.

Theorem 2 $D \vdash_{\mathrm{PD}} C$ *iff* $D \vdash_{\mathrm{RQ}} C$.

Proof: The proof from left to right results immediately from Theorem 1. Given $D \vdash_{\mathrm{PD}} C$, it follows from the definition of a procedural derivation that D, $C\Rightarrow \vdash_{\mathrm{RR}} \Rightarrow$, from which we obtain D, $C\Rightarrow \vdash_{\mathrm{RQ}} \Rightarrow$ by Theorem 1. But $C\Rightarrow$ abbreviates $\sim C$, and so we obtain $D \vdash_{\mathrm{RQ}} C$.

For the proof from left to right, assume that $D \vdash_{\mathrm{RQ}} C$. It follows from Theorem 1 that D, $C\Rightarrow \vdash_{\mathrm{RR}} \Rightarrow$. By the definition of relevant resolution, all members of D, $C\Rightarrow$ are used, so we need only show that the derivation of $\Rightarrow$ from D and $C\Rightarrow$ by relevant resolution can be converted into a procedural derivation. In order to do this we will need to show how to restructure the derivation so that $C\Rightarrow$ is used first, and also so that the derivation is *linear*, i.e., each step follows from a use of resolution which is applied to the previous step. This requires that we prove that the steps of a derivation may be reordered.

We illustrate how the steps of a derivation may be reordered in the special case of ground clauses (i.e., clauses which contain no variables). A careful proof is tedious but the following example should convince the reader.

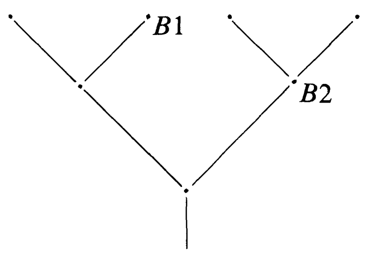
$$\begin{tikzcd}
C1 = B,A,L \Rightarrow R \arrow[dr, dash] & & C2 = L' \Rightarrow A,R' \arrow[dl, dash] & \\
& B,L,L' \Rightarrow R,R' \arrow[dr, dash] & & C3 = L'' \Rightarrow R'',B \arrow[dl, dash] \\
& & C4 = L,L',L'' \Rightarrow R,R',R''. &
\end{tikzcd}$$

Here clause $C4$ is obtained from $C1$, $C2$, and $C3$ by two uses of resolution. We may also obtain $C4$ from $C1$–$C3$ in a derivation where $C3$ is used in the first step:

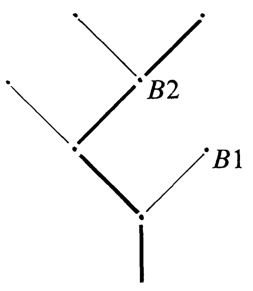
$$\begin{tikzcd} C1 = B,A,L \Rightarrow R \arrow[dr, dash] & & C3 = L'' \Rightarrow R'',B \arrow[dl, dash] & \\ & A,L,L'' \Rightarrow R,R'' \arrow[dr, dash] & & C2 = L' \Rightarrow A,R \arrow[dl, dash] \\ & & C4 = L,L',L'' \Rightarrow R,R',R''. & \end{tikzcd}$$

This does not establish that the steps may be reordered in derivations that contain variables. However, we may extend the result for ground clauses to clauses in general using the Lifting Lemma. (See, for example, [10], p. 211.) Take any relevant resolution derivation and let s be the collection of all substitutions of terms for variables used in the derivation. The result of applying s to each clause in the derivation is a ground clause derivation which can be reordered. Now reorder the original derivation in the same way. The result of applying s to this new ordering is the reordered ground clause derivation which has a case of resolution at each node. It follows by the Lifting Lemma that each node of the new ordering is a case of resolution.

Once we have shown that derivations by relevant resolution can be reordered it follows that a relevant resolution of $\Rightarrow$ from D, $C\Rightarrow$ can be restructured so that $C\Rightarrow$ is applied first. The derivation tree can also be put in linear form, for every case of nonlinear structure:



can be exchanged for a linear structure (by reordering branches $B1$ and $B2$ in the example given).



We now turn to the problem of refining our definition of *topic* and of showing that fragments of relevance logic obey the property that provable arguments contain only premises which belong to the topic of the conclusion. This property, remember, allows us to locate, for each conclusion, a subset of all the data which could possibly play a role in its proof. The results we have all concern the Horn clause fragment of RQ, that is, the set of theorems of RQ which correspond to Horn clause sentences. It is an interesting open question whether similar results hold for larger fragments of RQ.

With the preceding theorems in hand, the definition and proof are quite simple. The *topic* for a set of data given a conclusion C is simply the set of sentences which would be encountered in a full search for a proof of C using procedural derivation. More specifically, it is the smallest set of sentences closed under the following operation (assuming all sentences have been written in Horn clause form): The conclusion C belongs to the set, and when a sentence A in the set is such that $s(A) = s(A')$ for some substitution of terms for variables, and $L \Rightarrow A'$ is in the data, then the sentences $s(L)$ are in the set.

NOTES

1. Fodor ([4], p. 90) argues that the linguistic input system outputs the logical form of a sentence. However, it does not follow that the input system is responsible for drawing conclusions from the sentences so represented.
2. Strictly speaking, we are not describing modules here, but only kinds of modules, because we have said nothing about their data. For a module to be complete, it must satisfy two criteria: first, the data available must be comprehensive enough to entail anything that ought to be provable in its domain, and second, the rules must be strong enough to derive any conclusion that ought to be proved from the data. For the purpose of the examples discussed, we are simply assuming that the data are comprehensive.
3. The referee has pointed out that in this regard there is an important difference between the notion of a logical module and the idea of modularity in psychology. The neural structure of many organisms includes modules which are cut off from data which would be relevant if it were available. In psychology, it is almost a defining feature of modules that they are nonlogical. Our concern, however, is not so much whether logical modules actually exist in biological reasoners, but whether modularity is incompatible with logical correctness.
4. See, for example, [1], for a discussion of the importance of avoiding the Law of Contradiction, and of the application of relevance logic in doing so.
5. Actually, another way out is to adopt classical logic without negation, in which case contradictions cannot arise. However, we rule (fairly) that classical logic without negation is odd enough to merit the label 'nonclassical'.
6. This section was prompted by remarks on a previous version of this paper by Lee Bowie, to whom I am grateful.
7. Actually, we are looking for consequents that *match C*, but this is a complication that we need not go into for present purposes. (See the Appendix.)
8. We believe that the result can be extended from the negation implication and quantifier fragment RQ$-$ to the full system RQ of [8], for we are confident that RQ is a conservative extension of RQ$-$. Meyer reports ([1], p. 374) that R is a conservative extension of its negation implication fragment. We believe that his proof can be easily extended to handle the quantifiers, though we have not verified this.

REFERENCES

[1] Anderson, A. and N. Belnap, *Entailment*, Princeton University Press, Princeton, 1975.

[2] Cherniak, C., "Computational complexity and the universal acceptance of logic," *Journal of Philosophy*, vol. 81 (1984), pp. 739-758.

[3] Dunn, J. M., "Relevance logic and entailment," pp. 117-224 in *Handbook of Philosophical Logic*, eds. D. Gabbay and F. Guenthner, D. Reidel, Dordrecht, 1986.

[4] Fodor, J., *The Modularity of Mind*, MIT Press, Cambridge, 1983.

[5] Gentzen, G., "Investigations into logical deduction," *American Philosophical Quarterly*, vol. 1 (1964), pp. 288-306.

[6] Kolwalski, R., *Logic for Problem Solving*, North Holland, New York, 1979.

[7] Marr, D., *Vision*, W. H. Freeman, San Francisco, 1982.

[8] Meyer, R., J. Dunn, and H. LeBlanc, "Completeness of relevant quantification theories," *Notre Dame Journal of Formal Logic*, vol. 15 (1974), pp. 97-121.

[9] Newell, A. and H. Simon, "GPS, a program that simulates human thought," pp. 279-296 in *Computers and Thought*, eds. E. Feigenbaum and J. Feldman, McGraw Hill, New York, 1963.

[10] Robinson, J., *Logic: Form and Function*, North Holland, New York, 1979.

Department of Philosophy
University of Houston
Houston, Texas 77204-3785