LOGICAL LINKS AND

RELIABLE KNOWLEDGE REPRESENTATION

Phil Staines

Cognitive Science Research Paper

Serial no: CSRP 029

The University of Sussex
Cognitive Studies Programme
School of Social Sciences
Falmer
Brighton BN1 9QN

Current Address: Department of General Studies
University of New South Wales
Kensington, N.S.W. 2033
AUSTRALIA

LOGICAL LINKS AND RELIABLE KNOWLEDGE REPRESENTATION.

INTRODUCTION

The aim of this paper is to examine some aspects of the reliable representation of information in machine manipulated logic-like formalisms. While not restricting ourselves exclusively to it our focus will be on the representation of natural-language information - information we would normally and easily formulate in natural language - and its relation to the formalisms. We begin by drawing a parallel between the process of representing and using information in this way and the much older technique of symbolising the premisses and conclusion of an argument into a logical formalism to determine its validity. Serious doubts about this technique are raised via a proof that in its usual (unrestricted) application it is unreliable. This raises the question of the reliability of analogous knowledge-based information representation, inference and retrieval which we consider in detail. Finally, using an account of circumstances in which the logical technique is reliable, we are able to show under what circumstances we can have reliable knowledge representation in some of these formalisms. We consider in turn first order predicate calculus, production systems and prolog.

The technique of using logic to test the validity of arguments can be usefully considered as occurring in three stages. Firstly, finding in a suitable sentence form what the argument is; secondly, paraphrasing, symbolising or translating the argument into some symbolic notation; and thirdly testing the validity of the symbolisation and thereby (supposedly) deducing the validity or invalidity of the natural language argument.

Consider the third stage first. This stage, that of testing the validity of the symbolised argument is the one which motivates the other two. It is our facility, using logical techniques, for answering questions of the validity of symbolised arguments that leads us to attempt to use this method to test the validity of natural language arguments.

The first stage, often neglected, is that of identifying the argument. This includes not only identifying the explicit premisses and conclusion but also uncovering any implicit information - filling out the premisses in an enthymeme.

The second stage involves the crucial link between the natural language argument and its symbolised analogue. If the right relationship does not hold here then the validity or invalidity of the symbolisation will tell us nothing about the validity or invalidity of the natural language argument. This translation or paraphrase phase is the one we will be concentrating on in this paper.

Deductive question-answering programs which take information in natural language and answer natural language questions form perhaps the closest parallel with the logical technique of symbolising and testing for validity. The natural language input is 'translated' into a representation scheme in a database, a process corresponding to the symbolisation of natural language premisses in a logical formalism while the question is represented in the same scheme, a process correspo to the symbolisation of the conclusion and the system attempts to d it from its representation of the natural language informatio process corresponding to an attempt to show the symbolised argume valid. If the attempt to deduce succeeds the answer to the question be "yes" if the attempt fails the system may take (what is meant t the negation of the question representation and attempt to deduce If this succeeds the answer is "no" and if this fails then the a will be "don't know". If the system's designer is confident it has knowledge in the area and as we will later note has fully repres this information this latter process may not occur and failure to d will simply lead to a "no" answer.

The three stages identified above in the application of logic clear analogues here. The first stage, identifying the arg parallels isolating the information to be represented in the 'data (the premisses) and formulating the question (the conclusion) second stage, symbolising the argument, corresponds to translating information and question into the representational language. Whil third stage, testing for validity, is paralleled through the search deductive manipulation of that language.

In natural language deductive question-answering programs parallel with logic is obvious. However there is a signif parallelism in information representation systems generally. For mu the information we might attempt to represent in a machine-manipu formalism is for us best represented in natural language. So we h the translation process ourselves as programmers or as users . The stages are still there. Identifying the information to be represe representing it, and programming the machine to manipulate representation in such a way that we can interpret the repres result.

One case of this is data-base manipulation and query language person who has learnt to use one of these has learnt how to repr information ordinarily represented in natural language in this form and to formulate queries, naturally expressed as questions, i special query formalism. A less complicated case is a program wr to take a representation of a definite integral as input and to out representation of its value. In this case the representation is direct but even here we have a program that is intended to produ ouput that is a logical consequence of its input.

With this parallel drawn we now turn to showing the unreliab of the logical technique. Our argument will be that routine applica of the technique fail and that without an adequate general accoun the failure other applications are cast in doubt (without a shadow!

Since we are concerned with the use of logic to test the val of arguments in natural language we should be clear about the pro we are testing for - viz. validity. Although 'validity' is often more widely to refer to the general worth of an argument we wi concerned here with the following usual notion. An argument is vali and only if it is impossible that the premisses be true an conclusion false, that is, that the premisses logically imply conclusion. On the assumption of bivalence this definition implies valid arguments preserve truth - the conclusion of a valid argument be true when the premisses are. But it is worth noting tha consequence of the definition is that any argument with inconsi premisses is valid as is any argument with a necessary conclusion.

The purpose of the technique of symbolising and testing validity is to achieve a reliable indication of the validi invalidity or natural language arguments. To establish the unreliability of the method it is sufficient to find one clearly (in)valid argument that a routine application of the technique indicates is not (in)valid, Ne Mill consider three, two invalid arguments deemed valid and one valid argument deemed invalid, as each illustrates a different point to be made later.

Consider the following argument;

If John is in Sydney then John is in Australia.
If John is in Paris then John is in France.
So Either if John is in Sydney then John is in France
or if John is in Paris then John is in Australia or both.

This argument has true premisses and since neither disjunct is true a false conclusion. Consequently it is invalid. However a routine application of the techniques of logic e.g. symbolising it as

S -> A
P -> F
So (S -> F) v (P -> A)

deems it valid. Here we have symbolised 'if.then..,' by material implication represented by '->' and 'or' by weak disjunction given here by 'v' and abbreviated the English clauses to the corresponding letters. Had we chosen we could have given a more detailed analysis of the clauses using, say, a two place predicate in each of the component clauses, e.g. 'in(John,Sydney)' to represent the first clause and fifth clauses but the result would have have been the same. Truth-table methods (e.g.) reveal not only that this symbolisation is valid but that it would be still valid if either (but not both) of the premisses were missing.

As a second example of an invalid argument mistakenly judged valid consider the following [1];

If I have eternal life if I believe in Sod
then God exists.
It is not the case that I believe in God.
So God exists.

Agnostics may accept both premisses and reject the conclusion since the argument is invalid. However, were they to have faith in symbolic logic and use it to test for validity their faith (or lack of it) would itself be tested. A routine application of logic to this argument yields the valid symbolisation -

((B -> E) -> G)
-B
So G

What may be a clearer argument of the same form is - If I can levitate if .1 believe I can then I live in a belief-powerful universe. It is not the case that I believe I can levitate. So I live in a belief-powerful universe.[23

Examples of valid arguments deemed invalid are sometimes more contentious but consider the following;

If Kasparov wins then Korchnoi will not win.

It is possible that Kasparov will win.
So It is not the case that if Kasparov wins Korchnoi will »

Its routine symbolisation into modal logic using material implic for 'if..then..' remains invalid whichever (normal) nodal sys1 used.[31

This completes the argument that the technique of testinç validity is, in its routine application, unreliable. But what sc defence might someone anxious to defend logic mount? Two po« responses to the above argument are that firstly, despite appear the natural language arguments in the first two examples are valic in the third invalid, and secondly, that the arguments have been wr represented or symbolised.

In the first two .cases untutored intuitions are sufficient tc the onus of proof on those who claim that the arguments are (re valid.[4] A logician who has come to interpret his own use o^ English conditional as synonymous with the material conditional r able to reliably use logic to test the validity of (some of) his arguments but unless others use the conditional in the same way he not be able to test their arguments reliably.

The second criticism is that the wrong symbolisations have used. If a criterion of correctness of representation is that only arguments be deemed valid then of course it has to be conceded thai wrong symbolisation has been used. But this is a different clam the claim that an error has been made in the standard applicatic symbolic logic to the evaluation of arguments. The above symbolise are just routine applications of the technique, so that if the symbolisations have been used then the technique is wrong to recc them.

The unreliability of the technique makes its use on any parti occasion suspect. If it is known to fail and one is unat characterise and isolate the occasions when it does then one wil know whether this occasion is one in which it will also fail.

This pessimistic conclusion applies via the parallel deve above to logic-based natural language question-answering sys Consider the following information represented logically in a ques answering system:

If John loves Sally then he will marry Sally
If John loves Mary then he will marry Mary
It is false that if John loves Mary he will marry Sally.

Surprisingly, when asked if John will marry Mary the system is abl answer "Yes" and without makig any 'closed world' assumptions answer to the question "Does John love Sally" is "No". Intuitively course, given only this natural language information we would ex; "Don't know" answer to both questions.

A system susceptible to this sort of unreliability is E described in the work of L. Stephen Coles (Coles, 1972) . Et standing for ENGlish input, physical LAWs question-answering syste able to translate simple English input into predicate calculus fc well as store more complex hand translated sentences and deduct answer questions in English. Its information base was a compilati known physical laws and effects and its deductive inference performed by QA3.5 a resolution-based automatic theorem prover.

It is clear from the description (pp4B-53) that this system on a close parallel to the logical technique of symbolising and te for validity and so would also answer the above questions incorrectly.

Coles explicitly discusses the adequacy of first order logic as a representation for natural language, linking it to the following test for whether a person really understands a sentence "The true test, it seems, would be if he could satisfactorily answer all the "legitimately" answerable questions" that could be answered by anyone else who could be said to understand the sentence as intended by the speaker, but without regard to any particular universe of discourse."(p41). This leads him to define the epistemological adequacy of a representation of the meaning of a sentence in terms of the correspondence between the questions the sentence could be used to answer and the representations of the questions the representation of the sentences meaning could be used to answer. Taking the surface structure to describe the sentence and using the term 'deep-structure' for a meaning representation he offers the following definition. "...let S be a surface structure, T be a surface-to-deep structure mapping, D be the deep structure associated with S, and Q(X) be a function generating the class of legitimately answerable questions associated with X. Then if T(S) = D = (inv(Q))(T(Q(S))) both T and D are said to be adequate." (p42,43). "...where (inv(Q))(T(Q(S))) is larger than D then T is a filtering process, having deleted or failed to represent some legitimate aspect of S. When (it) is smaller than D, T is a noisy process, having injected some spurious information into the translation." (p43) Coles remarks that in computer implementations T is usually inadequate because it tends to "filter out the more subtle aspects of a sentence which are still the legitimate basis for questions and, a fortiori, meaning."(p43)

It is not clear what the terms "larger" and "smaller" mean here applied to meaning representations or deep structures. Since he has a first order logic in mind as the representation it is tempting to suppose he takes "larger than" to mean "logically implies but is not logically implied by" and "smaller than" to mean "is logically implied by but does not logically imply" However he remarks that "...T can easily be simultaneously inadequate for both reasons, i.e. filtering and noisy yielding a deep structure of comparable size but yet very different"(p43) This quotation reveals that he means that one representation can be both "smaller" and "larger" than another, not that T is sometimes noisy and at other times filtering.

Probably the scheme most faithful to his intentions is to compare the two classes Q(T(S)) and T(Q(S)) and take T to be filtering a particular S when the first class intersects but is not properly included in the second and to take it to be a noisy process when the second intersects the first but is not properly included in it. The upshot is that our example reveals that the translation scheme T that he adopts is noisy on some occasions. Taking S to be (the surface structure of) the conjunction of the premisses the two questions 'Will John marry Mary?' and 'Does John love Sally?' cannot be legitimately answered from S while their mappings can be from T(S). Put differently the question-answering system is unreliable, giving false 'yes' answers.

In general a "noisy" mapping is more misleading than a "filtering" mapping, since wrong answers are given. When the mapping is a filtering process typically the system generates fewer correct informative answers than could be generated from the natural language information represented in it. The questions in the shortfall are answered innocuously, if uninformatively, with "I don't know" However, if the system has been programmed to reply "No" when it cannot deduce an answer the situation is reversed. For as Coles remarks, the usual fault in computer implementations is that the mapping "filter(s) out the more subtle aspects of a sentence".

One deficiency of Coles analysis in terms of question answeri that he confines his discussion to legitimately answerable ques based on only ONE sentence. Ne finessed this difficulty above whe conjoined the three sentences from the database into one. However, the fact that a napping is adequate (in his sense) for two sent taken individually it does not follow that it is adequate for conjunction. For the set of legitimately answerable questions conjunction properly includes the union of the sets of legitii answerable questions of each conjunct in non-degenerate cases. neither conjunct logically implies the other there will be ques legitimately answerable from the conjunction that could not legitimately answered from either conjunct.C53 Consequently his n of adequacy does not suffice for the more interesting case in which question-answering system is e.g. using rules of inference with than one premiss, i.e. combining information in the database to d answers.

This deficiency could be handled using Coles concept legitimately answerable questions but a clearer approach is to us concept of Reducible consequence, and an 'inference engine' on representations. As before let T map surface structures of sent into some form of representation but take Q to be a function taking of sentences into sets consisting of the logical consequences of sets. Let Q' do the same relative to an 'inference engine' over the of representations. Then for a body of information specified by a s sentences, say I, we shall say that a representation scheme T tog with its inference engine (which determines Q') is DEDUCTIVELY REL if T(Q(D) * Q'(T(D). When these sets intersect we can say, still Coles' terms, that T is a filtering process relative to Q' over I the first set is not contained in the second and a noisy process the second is not contained in the first. Now when I is the set of sentences above about John, Mary and Sally we have shown that C transformation T, the usual process of representing sentences in order logic, is noisy with respect to Q' determined by an infe engine with the full power of the predicate calculus. We fill find this relativisation of the representation scheme to I and conceptually useful when we consider and contrast the reliabili other knowledge representation schemes.

Put in these terms we have argued that Coles' represent scheme is not reliable for some kinds of information routinely tak be within its domain, and the same argument was made, in effect, fc technique of symbolising and testing for validity. Inability on our to distinguish systematically between the I and members of Q(I) which T is reliable and those for which it is unreliable will lea unable on any particular occasion to be confident in the use of system. We now turn to a way of doing this for the standard cs symbolisation into first order logic. We ^re then able to take treatment as a fixed case for comparing other knowledge represent schemes * specifically production systems and horn clause logic.

Our approach will be initially to constrain the domain of T class of sentences where it is deductively reliable but then proce show that despite this reduction in scope in one area there i expansion in scope in another because sentences generally thought outside its domain can also be mapped reliably by it. In orde restore reliability we need to distinguish a number of cases. Fi those where the symbolisation is valid need separate treatment those where it is invalid. We revert to the symbolisation techniq the exposition of these ideas.

There are a number of different interpretations given to a logic symbolisation.[6] Since it enables the clearest simple description the issues we will work with the case in which the symbolisation treated as a genuine language and thus as consisting of sentences whi will be genuinely true or false. Kalish and Montague (1964) give t clearest introduction to this approach which is best seen as augmenti a natural language with additional symbols and constructions augmenting written English with parentheses and with some tru functional connectives e.g.'-', '&',' v', '->', etc. For example '(is prime) -> (13 is not divisible by 7)' is a true sentence in Kali and Montague's language of symbolisation.

Unlike some other interpretations of symbolisation t symbolisation is not a form but a concrete sentence. (although it wi have a form) In order to speak generally about sentences of English a sentences in the English-like symbolisation we shall use the variabl 'X','Y' and 'Z' for sentences of English and the lower case 'x', 'y' a 'z' for sentences of the symbolisation, leaving it to the context determine whether these literally stand for or (simply) refer to t corresponding sentences. Note that there is no assumption that the sentences are unstructured. We also use the double double arrows '<== for logical equivalence and the single double arrows '=>' and '<=' f logical implication, and take one sentence to logically imply anoth when it is impossible that the first be true and the second false.

VALID SYMBOLISATION

The tradition usually requires that a sentence X and i symbolisation x be synonymyous or at least logically equivalent - th they have the same truth conditions. Where this does hold the validi of a symbolisation establishes the validity of the argument symbolise So when

```
        English       Symbolisation

           X      <==>     x
           Y      <==>     y
          ---             ---
     So    Z      <==>     z
```

the validity of the symbolisation establishes the validity of t English. However since frequently this relation does not hold (e.g. c earlier discussion establishes that it does not hold between sc instances of 'If X then Y' and its usual symbolisation '(X) -> (Y)' we will need to find some other relation that suffices for a val symbolisation to establish the validity of an argument. If we weaken t relation to logical implication and distinguish premisses fr conclusion (and valid symbolisations from invalid ones) then t following relation between an argument and its symbolisation sufficient for the latter to validate the former:

```
        English       Symbolisation

           X       =>      x
           Y       =>      y
          ---             ---
     So    Z       <=      z
```

This follows from the transitivity of logical implication and preservation under conjunction. Since the symbolisation is valii conjunction of its premisses logically implies its conclusion. But conjunction is implied by the conjunction of the English premisse to via transitivity it follows that the English premisses logi imply the English conclusion, and hence that the natural Ian argument is valid.[73

Consequently, if we confine ourselves to symbolisations in the premisses are logically implied by the English premisses a which the conclusion logically implies the English conclusion we ca confident that the validity of the symbolisation will be a rel guide to the validity of the natural language argument. (although necessarily its invalidity) Armed with the knowledge of these (we conditions we need no Longer be suspicious of ALL attempts to valid symbolisation in which there is not equivalence between the Englis its symbolisation. The weaker relation of logical implication sometimes suffice.

The most important application of this concerns the symbolis of the natural language indicative conditional. We can ide arguments in which, despite their inequivalence, statements in En of the form 'If X then Y' can be represented by statements of the '(X) -> (Y)'. For although much is controversial about the analysi the indicative 'If...then...' it is uncontroversially false whe antecedent is true and its consequent is false and since this is only combination in which the material conditional is false impossible that the English indicative conditional be true and corresponding material conditional false. Hence we have

If X then Y •> (X) -> (Y).

While there may be 'joke' conditionals which *are* equivalent to material conditional symbolisations (e.g. If Tulloch wins then monkey's uncle) in serious cases, like conditionals used to ex causal relationships, it is very difficult to make out a case fo stronger relation than logical implication. Such conditionals foi sizable proportion of the information to be represented in some e systems.

Thus where arguments of the Modus Ponens form in English are valid by their logical symbolisations

```
        English              Symbolisation

          X           *>          X
        If X then Y   *>       (X) -> (Y)
       ------------          -----------
   So     Y           <*          Y
```

arguments of the form of the paradoxes of material implication are shown to be valid by their valid symbolisations. Thus in

```
        English              Symbolisation

          Y           =>          Y

    So If X~then Y    =>       m" -> (Y)
```

although the symbolisation is valid it does not establish the val of the English argument by the above sufficiency conditions since the required relation does not hold in the conclusion.

Having seen this it might be tempting to describe the requirements for reliable representation of conditionals simply as: they must not occur in the conclusion if the symbolisation is valid. But after looking at two other applications we shall see that matters are considerably more complicated.

Sometimes conjunction is used in natural languages to convey the information that events described in the conjuncts occurred in temporal order. When this information is represented by truth-functional conjunction the symbolisation is logically implied by but not logically equivalent to the information it represents. Similarly, although strong disjunctions can be represented logically equivalently in truth-functional logic it is sometimes argued that there is an intensional (use of) disjunction which cannot. Such disjunctions, say 'X or Y', it is argued, support the corresponding subjunctive conditionals 'If it were not the case that X it would be the case that Y' which truth-functional disjunctions (whether strong or weak) do not. Such disjunctions nevertheless logically imply weak disjunction.

It is notable in the above examples that we have used the upper-case letters 'X' and 'Y', which stand for unsymbolised sentences, on the right hand in the above as symbolisations. This is to indicate that these sentences or clauses contain no parts which belong only to the language of symbolisation - they are pure English rather than a mix of English and other constructions of the language of symbolisation (e.g. the special connectives). In other words these sentences and clauses are viewed as symbolised, vacuously, as themselves. This procedure ensures that the English sentence or clause and its symbolisation are logically equivalent. This is important because the situation is frequently different. Often the component clauses of a sentence are not themselves symbolised by something logically equivalent. Thus just as a conditional sentence is routinely symbolised by something logically weaker than it so a conditional clause is routinely symbolised by something weaker than it. However that the clause is weaker does not always ensure that the sentence it is part of is too. Thus although English conditionals logically imply their corresponding material conditionals, we have the following relation if they are embedded in the context 'It is not the case that...'

It is not the case that if X then Y   <=   -((X) -> (Y))

In view of this, if a component of a sentence is being represented by a non-equivalent construction we will need to consider the kind of context it is in. For elementary logic we consider the following common contexts; negation, conjunction, weak disjunction and the antecedent and consequent of a conditional. We can offer the following conditions under which implication relations can be determined for embedding contexts. However, due to both the infinite extent of English and the vagaries of its usage, while offerred confidently they nevertheless remain open to review in context.

A sufficient condition for

1. It is not the case that X to logically imply -x is that x => X.
2. X and Y to logically imply (x) & (y) is that X => x and Y => y.
3. X or Y to logically imply (x) v (y) is that X => x and Y => y.
4. If X then Y to logically imply (x) -> (y)is that x => X and Y => y.

The most notable of these requirements is for the conditiŏn* While the symbolisation of its consequent must be at least logical implied, the symbolisation of its antecedent must logically imply i antecedent. Just as we have asymmetric conditions for the premisses < conclusion of an argument so we have asymmetric requirements for 1 antecedent and consequent of conditionals.

The failure of the second argument illustrating the unreliabil: of logic can now be explained, as can the John, Sally and Mary knowlet representation failure. In each of these cases although the or conditionals occurring occur in the premisses (or their analogue) st of them occur in contexts which do not ensure that the statements tl contain them logically imply their symbolisations. In the first c< this context is the antecedent of a conditional and in the second it inside a negation operator. Thus using B, E and 6 to abbreviate i relevant statement components or clauses (see above), although we havi

If if B then E then 6 => (if B then E) -> (6),

we do NOT have

If if B then E then B *> ((B) -> (El) -> (6).

In the second case too the symbolisation is not logically implied but this case it logically implies the English,

For a fuller account of the circumstances of relial representation we cite a result which enables us to systematical exploits these asymmetries.[83 To understand it we need to note that argument is valid if and only if the set of statements consisting of : premisses and the negation of its conclusion is inconsistent and we mi introduce the concept of a purely positive and a purely negat: occurrence of a clause or statement component in a statement.

Where the language of symbolisation includes the truth-*unctior connectives '-', 'v', 'I' and '->' we shall say that a clause statement component of a statement (henceforth a component) has a pun negative occurrence where the only connectives it lies in the scope are those just mentioned and the sum of the number m of '-' ' s it 1: in the scope of and the number n of '->' 's it lies in the scope of 1 antecedent of is odd. If this number (possible zero) is even and thi are the only connectives it lies in the scope of we shall say it hai purely positive occurrence. Thus in

((it rains) -> (it floods)) -> (-(the road is safe))

the component 'it rains' has a purely positive occurrence being in 1 antecedent of two '-' 's and in the scope of no '-' 's as has '-(• road is safe)' and the whole statement itself, 'it floods', 'the road safe' and '(it rains) -> (it floods)' have purely negative occurrence!

The following result can be proved:

a. If in an inconsistent set of statements any purely positivi occurring component is replaced (at that occurrence by a statement th« logically implies it the resulting set is inconsistent. b. If in inconsistent set of statements any purely negatively occurring componi is replaced by a statement that it logically implies it the result: set is inconsistent.

Paraphrased to apply to arguments we have:

*) Any argument obtained from a valid argument by replacing - any purely positive occurrence of a component in the premisses by a statement that logically implies it or any purely negative occurrence in the premisses by a statement that it logically implies or any purely positive occurrence in the conclusion by a statement that it logically implies or any purely negative occurrence in the conclusion by a statement that logically implies it - will also be valid.

We should note of course that when a component and its symbolisation are logically equivalent these distinctions need not be made, since in this case each implies the other.

These results enable us to say quite generally under what circumstances we can reliably symbolise a statement component by something it implies or something which implies it. Specifically, we have implicitly specified circumstances in which the routine symbolisation of 'if..then..' clauses can be reliably performed. We have also enabled a considerably broadening of the reliable range of application of the technique to non-routine symbolisations.

Perhaps the best way to iillustrate the wider applicability of these asymmetric relations is to use a truth-functional symbolisation to show the validity of an argument in which some of the connectives are not truth-functional. The argument

Either Smith died because he drank alcohol while he was on penicillin or Smith needs quinine because he has malaria. (But) Smith did not drink alcohol while he was on penicillin. So Smith needs quinine.

is shown valid by the symbolisation

(Smith died & Smith drank alcohol while he was on penicillin) v (Smith needs quinine t< Sith has malaria). -(Smith drank alcohol while he was on penicillin). So Smith needs quinine.

in which 'because' is symbolised by the truth-functional '&'. Since the 'I' clauses have (i) a purely positive occurrence in (ii) the premisses of (iii) a valid symbolisation the argument obtained by replacing these clauses by the corresponding 'because' clauses, which imply then), is also valid.

Another extension, specifically for predicate logic involves the symbolisation of sentences of the forms

Most A are B, Many A are B, A few A are B, Several A are B

by the forms[9]

(Ex)(A(x) *I* B(x)) and (x)(A(x) -> B(xM b (ExMA(x)).

Some authors, Guttenplan and Tamny (1978, p71) are two, advocate translating sentences of these forms into sentences of the form 'Some A are B' which can be represented in the language of the predicate calculus by statements of the first form. As a piece of GENERAL advice this is a mistake since sentences of these forms logically imply but are not equivalent to 'Some A are B'.

Thus the following invalid argument symbolised according to this technique

```
      English                               Symbolisation

   All dogs are mammals            =>   (x)(dog(x) -> mammal(x))
   Some dogs are poodles           =>   (Ex)(dog(x) & poodle(x))
 -------------------------            --------------------------
So most mammals are poodles       =>   (Ex)(mammal(x) & poodle(x))
```

has a valid symbolisation. The conditions are not satisfied in
conclusion. In contrast the following argument represented analog
is shown to be valid since the conditions hold.

```
      English                               Symbolisation

    All dogs are mammals            =>   (x)(dog(x) -> mammal(x))
    Most dogs are carnivores        =>   (Ex)(dog(x) & carnivore(x))
  --------------------------            ----------------------------
So  some mammals are carnivores  <=    (Ex)(mammal(x) & carnivore(x)
```

While there is no natural logically equivalent form of representati
the predicate calculus we can take advantage of the fact that in ge

(Ex)(A(x) -> B(x)) <= Most A are B <= (x)(A(x) -> B(x)) & (Ex)(A

to specify conditions under which statements of the form 'Most A ar
can be reliably represented in the predicate calculus. Where the fo
the left is used to represent it in purely positive contexts in
premisses or in purely negative contexts in the conclusion and wher
form on the right is used to represent it in purely negative contex
the premisses and purely positive contexts in the conclusion
(assuming of course that the other components of the argument
adequately represented) it will be valid if its symbolisation is.

RELIABLE KNOWLEDGE REPRESENTATION (i) FIRST ORDER LOGIC

These conditions for reliable use can now be used to show
what circumstances some forms of natural language information c
reliably represented in various forms of logic-like notation. In vi
the asymmetries noted we will need to distinguish within the doma
the mapping between the information that is being represented
premisses) and the information being obtained (the conclusion or a
to the question).

Before looking too generally let us see how these conditions
be used to enable the reliable representation and deductive retriev
statements of the form 'Most A are B' in part of Coles' ENGLAW sy
We will be able to use this example to draw some general princ
applicable to other systems of representation.

It is essential to what follows that Englaw give a 'Don't
answer when it is unable to deduce the representation of a question
its information store. We are able to achieve reliable representati
'most A are B' clauses if we complicate the mapping T (between En
sentences and Predicate calculus) in the following way. We
distinguish its operation in representing information for the data
from its operation in representing statements corresponding to na
language questions.

When representing clauses of the form 'Most A are B'

A) For representation in the database i) map those cl
routinely mapped into a purely positive occurrence into componen

the form (Ex)(A(x) & B(x)) ii) map clauses routinely mapped into a purely negative occurrence into components of the form (x)(A(x) -> B(x)) & (Ex)(A(x))

B) Question representation i) map those clauses routinely mapped into a purely negative occurrence into components of the form (Ex)(A(x) & B(x)) ii) map clauses routinely mapped into a purely positive occurrence into components of the form (x)(A(x) -> B(x)) & (Ex)(A(x))

This readily applicable mapping routine allows a reliable extension of both the standard domain for ENGLAW and the questions it can handle. Two simple augmentations include the ability to handle questions of the form 'Are most A B?' and to represent information of the form 'It is not the case that most A are B'. The full range is inferrable from the above conditions.

It does however ensure that the mapping T is a filter (see above).i.e. that that there will be questions answerable from the natural language information that are not answerable by the system from its representation. By way of illustration, the information that most A are B and most A are C permits the question 'Are some B C?' to be answered affirmatively, where ENGLAW would return a 'Don't Know'. We will see shortly that in this form it is even more restrictive.

The situation is similar but much more restrictive for reliably representing natural language indicative conditionals in knowledge representation schemes based on first order logic. For while clauses of the form 'Most A are B' both logically imply and are logically implied by (different) clauses expressible in the predicate calculus and so can be reliably represented in both purely positive and purely negative contexts as we have seen, there is in general only one reasonable candidate for representing the indicative conditional namely the material conditional since this is logically implied by the corresponding indicative conditionals but does not in general imply them. The result is that on present information we are only justified in expecting general reliability when representing clauses of the form 'If X then Y'

A) For representation in the database i) map those clauses routinely mapped into a purely positive occurrence into components of the form 'X -> Y'.

B) Question representation i) map those clauses routinely mapped into a purely negative occurrence into components of the form 'X -> Y'.

These are very restrictive conditions, limiting as they do the general domain of application of the mapping T. But the alternative, if we are to use first order logic and the usual method of representation, is unreliability - something which may be a life-and-death matter when, say, a knowledge representation system is being used.

The requirements can be broadened slightly if we are able to identify a class of conditionals which are true when their antececent and consequents are both true. (A number of non-standard logics for the conditionals include this condition) For these conditionals we have

X & Y => If X then Y => X -> Y

and so we can reliably represent them by 'X & Y' in negative contexts in the database and in positive contexts in the question representation.

Before looking at other representation systems we should examine one highly restrictive aspect of this approach that the reader may have noticed - an aspect that is much more restrictive in general knowledge representation than in the logical technique of symbolising and te the validity of arguments.

As we have seen, when a statement or clause is not symbolise represented by something logically equivalent to it we need not ab the whole venture. In those cases where the relation is the weaker of logical implication we must note where it occurs in the argume knowledge representation system if we want to be assured of reliab However an important difference between the ventures of arg symbolisation and knowledge representation is that while in the f case the symbolisation can be tailor-made for a particular arg (inference) in the latter case the tame representation in the data will be used by the inference engine for many different inferences.

The 'symbolise and test for validity' technique is us expounded in conjunction with a labour saving recommendation tha symbolisation should be as simple as possible, put metaphorically b philosopher W.V.O. Quine in his maxim of shallow analysis: 'Whe doesn't itch, don't scratch'. Although this is bad advice determining invalidity it is useful practical advice when symbolisation is valid. When translating into a logical symbolism advice is to translate no more structure than is necessary fo logical techniques to show the symbolisation is valid. Thus i argument has the Modus Ponens pattern 'If X then Y, X so Y' then is no point in uncovering the structure of X or Y - the technique logic will show that 'X -> Y, X so Y' is valid and that is sufficie

While this maxim is merely labour saving when the components untranslated could be translated into something logically equivale is essential when they cannot. For 'oversymbolisation' may result symbolisation that does not satisfy the weakest conditions we have give us a reliable guide to the validity of an argument. Were w symbolise X from the previous paragraph by a symbolisation it logi implies, say x, we would have no guarantee that the first premiss adequately symbolised, since x has a purely negative occurrence i If we were to translate it by something which logically implied it the second premiss would be unreliabily symbolised. Of course translated it differently in each premiss then we might no longer formal validity (or even actual validity) and we certainly woul have a symbolisation of the Modus Ponens form.

A dramatic example of this is that an argument 'If X then Y X then Y' is not shown valid by the symbolisation 'X -> Y so X - Too much structure has been uncovered and the conditions no longer in the conclusion. It can however be shown valid by symbolisi trivially as 'Z so Z' where 'Z' abbreviates 'If X then Y'. Had however used this technique for the Modus Ponens example abov symbolisation would not have been shown valid by logical techn (being formally invalid).

Here we suit the level of analysis to the specific infer However in using logic as a knowledge representation scheme we d have this flexibility. Our symbolisation or representation is used for all. So we are apparently faced with the dilemma of being unab answer some questions in Q(I) either because we have uncovered too structure and cannot apply the adequacy conditions to ensure reliab or because we have not uncovered enough structure for the infe engine to use in its input.

One approach to a solution is to let T be one-many and several representations for any sentence whose struc representations are not equivalent to it. So, for example, a condit 'If X then Y' could be both represented in a logically structure non-equivalent way by 'X -> Y' and in a logically unstructured but logically equivalent way by itself (or coded abbreviation). In the latter case the rules of inference would treat the clause as unstructured. There would be no need for more than one representation where conditional clauses are mapped into purely negative contexts in the data-base as the constraints on T for reliable representation exclude representing them by material implication.

With different motives Coles (1972) included two representations for every sentence in ENGLAW. One in English and one in predicate calculus. The indexed English text was to enable quick answering of descriptive questions asking e.g. for a description of a law, while the predicate calculus text was used to deduce information. However in his system the two representations do not interact in the way we have suggested.

Deductive question-answering systems using the full predicate calculus are, as we have seen, unreliable in their unconstrained application. They are also not as widely used as it once seemed they might be. While this is fortunate for those concerned with reliability it would be wrong to take the former to have caused the latter since the unreliability is not widely appreciated. The reason is more plausibly attributed to an inability to avoid the combinatorial explosion through a failure to adequately direct inference. As we shall now see the weaker logic-like representations of production systems fare considerably better in reliable knowledge representation.

RELIABLE KNOWLEDGE REPRESENTATION ii) PRODUCTION SYSTEMS

Production systems form the main knowledge representation and manipulation component of a number of recent expert systems and since much of the information they represent is naturally expressed by conditionals in English we are prompted to consider their reliability. Although production systems form a diverse group we shall briefly sketch their structure and then look at the reliability of one main kind.

Production systems are taken to consist of three parts; a rule base or (ordered) set of rules, a database, short-term memory buffer or context and an interpreter. The heart of production systems is the rule base. This is a collection of ordered pairs, frequently described as conditionals because of an analogy with their role in Modus Ponens. Indeed the Handbook of Artificial Intellegence (p190, Barr and Feigenbaum) puts it: A production rule is a statement cast in the form "If this condition holds, then this action is appropriate". Following suit we shall use the terms antecedent and consequent for the two components of the ordered pair. When the antecedent is found or matched in the database the action specified in the consequent is carried out and the production (rule) is said to have fired.

A simple database is a set of unstructured symbols, some of which will typically match the antecedents of some of the productions, but can also be quite a complicated data structure. However as Davis and King note (Davis and King, p303) "Whatever the organisation of the data base, one important characteristic should be noted: it is the sole storage medium for all state variables of the system....There is nothing but the single data base, and all information to be recorded must go there."

The key function of the interpreter is to decide which of the productions could be fired by comparing antecedents with the database and then choosing which one to fire first. When the production system is backward chaining, as in MYCIN, the interpreter starts with a consequent, handling the process of finding a production with that consequent and looking in the data base or the consequent of other production rules (or both) for a match with its antecedent, repeating.

To introduce a discussion of reliability we shall consider w Winston (Winston, p148) calls 'simple deduction oriented' productio The situations that trigger or fire these productions are 'specif combinations of facts'(p144) and 'the actions are restricted to be assertions of new facts deduced directly from the trigger combination."(loc. cit.) The simple ones determine a single conseque We can think of these simple facts as being represented by unanaly sentences or simple predicate-argument forms and their negat analogues. The data base can be viewed as a set of these. Antecede contain simple facts or conjunctions (analogues) of them.

We can think of these representatives or analogues as being truth-functional negation and conjunction and think of the product rule as expressing material implication, if we like, since the infere analogue performed by the system is consistent with this interpretati This is in fact how they tend to be thought of and described by system designers on occasion and in some cases the system is program to translate them back into English conditionals, conjunctions negations when explanations are called for. However, the logi behaviour of the system is so limited, the logical links are so we that a number of other construals are also possible.

What is the logical behaviour of production systems of this so After remarking that a rule can be viewed as a simple conditio statement Davis and King continue (p301) "and the invocation of ru (can be viewed) as a chained sequence of modus ponens actions." Th is,however, no need to look at this chain link by link since we simply collect together all the representations used as "premisses" any stage in the chain, or for a smaller set all those used initia (since there may be redundancy) only as premisses. The result, in eit case, is an inference in which the premisses consist solely of sim facts and conditionals with simple facts as consequents and simple fa or their conjunctions as antecedents. The conclusion is restricted being a simple fact (as we are calling these representations). With connectives interpreted truth-functionally these arguments are valid.

We can finally ask if production systems of this kind do relia represent natural language information. The answer will naturally dep on what we take to be the domain of the mapping T. Since the o connectives are conjunction, negation and the conditional let us firs assume that conjunction and disjunction are equivalently symbolis Then if T is constrained to the sentences that can be routinely map into such a system the representation will be reliable, by the gene conditions given above for the full predicate calculus since the o conditionals represented are in purely positive (in fact of depth occurrences in the database. None will occur, for example, in conclusion or embedded in the antecedent of another conditional.

While this constraint will filter out a routine rendering natural language conditionals in all but reliable contexts, the limi expressive and deductive potential of production systems is not pr against someone trying non-routine representations. Consider troublesome first premiss of the second argument cited in this paper.

If if B then E then G

Some production system enthusiast, viewing production rules as mater implication, might observe that (B -> E) -> G is equivalent to (-B -> & (E -> G) and hope to represent this information in the correspond pair of productions - but presumably common-sense would prevail to prevent this process when the question of the truth of the first production arose.

A more likely source of unreliability in such systems is the failure of the production system (and truth-functional) conjunction to express the notion of temporal sequence that is sometimes conveyed by natural language conjunctions. Consider

If John marries and has children then his mother will be pleased

Conditionals like this may well not retain their truth value when the conjuncts in the antecedent are interchanged. In cases like this the conjoined antecedent will logically imply but not be logically implied by its truth-functional representation and since it is in a negative context in a conditional the representation will be suspect. The database may be successively augmented by the representations of 'John has children' and 'John marries' permitting the firing of the conditional and apparently endorsing an unjustified natural language inference. These remarks apply of course to stronger and more expressive representation schemes as well, like the predicate calculus.

Production systems form a diverse collection (See e.g. Davis and King) and they cannot all be as easily connected to a logical or argument model as simple deduction-oriented production systems. In what follows we comment briefly on the reliability of some systems that can be compared with a model of this kind.

In some production systems rules are used with consequents which subtract simple facts from the database rather than adding to it.[10] In these systems the order of firing of the productions is typically important (although not always). However reliability is not lessened since the effect of these rules, without the systems being further complicated, is to restrict what can be inferred.

Some systems have productions whose action has effects outside the database. One kind are those which add productions to the rule base. These can be viewed as conditionals with conditionals in their consequents and as such pose no threat to reliability since both the embedding and the embedded conditional occur in a positive context in the premiss of the analagous argument. A similar treatment can be given to productions which activate (sets of) productions. Productions which deactivate pose no threat either since they restrict (caeteris paribus) what can be inferred. This is an effect strongly contrasted with asserting the denial of a conditional as we have seen above.

More complicated systems include those in which the antecedent includes arbitrary truth functions. Taking conjunction and negation as the only connectives, augment the above simple deduction-oriented production rules so that in the antecedent conjunctions may appear within the scope of negations (and, as before, other conjunctions) and negations of compounds may appear within the scope of conjunctions. A production rule may be fired when it is evaluated as true relative to the database. That is, when the conjunction of facts in the database logically implies it. Concerning reliability, the same remarks apply as for simple deduction-oriented production systems, with the additional risk that the augmented expressive power of the antecedent may encourage its use to express forms of natural language which could otherwise have not been expressed or only expressed with difficulty. In such systems the temptation may arise for example to represent 'If if B then E then G' by a production of the form

-(B t -E> -> B

and this would be ready to fire if the database included '-B'. However, when the other components are tquivalently represented it follows from the general conditions for reliability that unembedded conditionals nay be reliably represented as productions while conditionals embedded in the antecedent of conditionals may be reliably represented by the usual denied conjunction when it occurs in a negative context within the antecedent of the production.

RELIABLE REPRESENTATION iii)HORN CLAUSE LOGIC (pure Prolog)

With the increasing use of prolog as a knowledge representation language the question of its reliability in this use becomes more important.[11] Horn clause logic can be thought of as the basis of the programming language prolog. Although the usual implementations of prolog have considerably greater expressive potential than the horn clause core, at least on the surface (i.e. in relation to routine translations), the reliability of the extensions presupposes the reliability of their horn clause basis. He confine our remarks to this restriction which we call pure prolog.

Initially to facilitate comparison we limit ourselves to horn clauses in propositional logic with letters taken as abbreviating unanalysed propositions. Call these letters and their negations literals. A horn clause is then a disjunction of literals with at most one unnegated literal. He can refer to the unnegated literal as the head and the disjunction of negated literals as the body and following Clocksin and Hellish (1981) call those clauses with a head 'headed' and those without 'headless'. If we restrict conclusions to being negations of headless horn clauses and restrict premisses to being (conjunctions of) headed horn clauses we have what can be identified as horn clause expressible inferences.

These clauses correspond in the premisses to pure prolog programs consisting of rules of the form 'A :- Bl,B2,...,Bn.' (n>0) and facts of the form 'C.' where all literals are unnegated. The conclusion corresponds to questions or goals of the form '?- Dl,D2,...,Dm.' <m>(9) again with all literals unnegated. These clauses correspond to premisses of the forms '(Bl & 62 & ...&Bn) -> A' and 'C and conclusions of the form 'Dl fc D2 & ... It Dn', since in the latter two cases the denial of a disjunction of negated literals is logically equivalent to the conjunction of unnegated literals.

This correspondence indicates the constraints on the routine symbolisation into horn clause logic. The main points being that structurally represented conditionals are excluded from all but positive occurrences in the program or premisses and are excluded from the conclusion. On the assumption that conjunction has been equivalently represented we can then be assured of the reliability of this representation by the adequacy conditions when prolog returns a "yes" answer and the argument is valid. However as we will shortly see we do not have a parallel assurance of invalidity when pure prolog returns a •No" answer.

So the only risk, barring equivocation, of unreliability of a "yes" answer for routine representations is non-equivalent representations of conjunction. The same remarks apply here as for production systems.

Hhat then of non-routine symbolisations? These are confined by the very limited means of expression available. . For example, disjunctions of positive literals cannot be expressed in program or goal. And while simple bi-conditionals can be expressed as two rules, as Kowalski (1979, p202) has observed horn clauses often express only the if halves of iff definitions. His specific worry is their inability (in general) to express conditionals with conditional antecedents. The most plausible candidate is the conjunction of conditionals noted above (production systems), but the second conditional, having a negated literal in its antecedent but not in its consequent cannot be expressed in a horn clause. We can observe that this lack of expressiveness is a key feature of their reliability.

Kowalski's main concern here is the expressiveness of quantified horn clauses. We take the literals to also include predicate-argument forms in which the arguments can be constants or variables (or functors [illegible] The corresponding e.g. to symbolising sentences like 'All A are B' as '(X)(A(X) -> B(X))' or in this form of horn clause as '(X)(-A(X) v B(X))' and in the prolog notation of 'b(X) :- a(X).' Sentences of this form are suspect in full logic because they are often only logically implied by and not equivalent to the information expressed by the sentences they symbolise.

In particular it is suspect when it occurs in a negative context, like the antecedent of a conditional, in the premisses of a valid symbolisation. Thus a routine symbolisation of the argument

```
      If all John's students are happy then John is happy
      John has no students
     ------------------------
   So  John is happy
```

will be valid but fail to show the validity of this argument because the first premiss does not logically imply its symbolisation.

In an analogous fashion to the conditional, pure prolog avoids these unreliable contexts by restricting what is routinely translated in this way to purely positive contexts in the premisses.[12]

SYMBOLISATION INVALID.

When a deductive question-answering system cannot deduce the answer to a question-representation two considerations are frequently discussed: firstly, is the inference procedure complete and secondly under what assumptions can failure to deduce be taken to imply that the answer is false. In this section, rather than addressing these questions directly we attempt to answer a question presupposed by them. This question is the compliment of the one we have been considering. When is failure of logical implication in the representation mirrored by a corresponding failure in what it represents? Put in logical terms under what circumstances does an invalid symbolisation ensure the argument it symbolises is invalid.

The superficially simple business of establishing the invalidity of an argument is surrounded by a minefield of misconceptions. So it may be best to begin with a sequence of claims designed to counter some of these. The beginning and end of the list may only be misconceived by few but some elements of the list are misconceived by many.[13]

i) That a proof of the conclusion of a symbolisation in a certain system has not been found does not always imply that it cannot be found.
ii) That a proof of a symbolised conclusion cannot be found in a logical system does not always imply that the symbolisation is formally

invalid.

iii)That the symbolisation is formally invalid does not always imply that the symbolisation is actually invalid.

IV) That the symbolisation is actually invalid does not always imply that the argument it symbolises is actually invalid.

v) That an argument is actually invalid does not always imply that its conclusion is false.

The crucial asymmetry with validity occurs in the centre of this list. Barring equivocation, if a symbolisation is formally valid then the symbolisation is actually valid, since by definition every argument of a valid form is valid. But if a symbolisation is formally invalid it does not in general follow that actual arguments having that form are invalid. For example the argument form, called affirming the consequent is an invalid form, but there are arguments of that form that are valid. Put another way, not every argument with the form

```
 D
 P -> Q
-------
 P
```

is invalid since in a specific case Q may logically imply P and so that concrete argument will be valid.

The general problem, exemplified here, is that forms may fail to formally express logical interdependencies between components of an argument, and while unexpressed logical interdependencies cannot affect an argument that would be valid without them, they can crucially make "apparently" invalid arguments (arguments having an invalid form) actually valid. Consequently the techniques of formal logic while useful for determining the invalidity of forms of argument have limited application in determining the invalidity of actual arguments.

One relatively minor exception which serves to reinforce this point is the class of contravalid argument forms. These arguments have formally necessary premisses and formally inconsistent conclusions. An example is

```
    P v -p
-------------
    q i -q
```

Every concrete argument of this form is invalid.[14]

Since the techniques of inference in machine implementations are formal these remarks have some rather restrictive consequences for determining that represented information does not follow from other (inevitably formally) represented information. Thus while few people make the mistake of thinking that a 'No' answer from a prolog program implies (on its own) that the assertion is false, the usual view is that the answer can be interpreted as meaning that the represented information does not follow. Without further assumptions the most that can be inferred is that the corresponding argument form is invalid. Formal invalidity does not in general ensure actual invalidity.

Suppose nevertheless that despite the failure of the usual tools for determining invalidity, a concrete symbolisation has been judged invalid. Under what circumstances can the invalidity of the argument it symbolises be reliably inferred? The conditions are

```
            (Symbolisation invalid)
        English          Symbolisation

           X       <=       x
           Y       <=       y
          ---              ---
    So     Z       =>       z
```

which are the mirror image of the conditions for validity. If the symbolisation is invalid then it is possible for x and y to be true and z to be false i.e. for x and y and -z to be true. Hence it is possible for any combination of statements logically implied by these to be true. So it is possible for X and Y to be true and Z to be false, since by contraposition for logical implication -z logically implies Z is false. Hence the English argument is also invalid when these conditions hold.

Thus where we do have an invalid symbolisation of the form of affirming the consequent (although we may have trouble showing it to be invalid) we will not know if the argument it routinely symbolises is itself invalid, since the conditions are not satisfied. Thus in

```
            English           Symbolisation

               Y            <=        Y
          If X then Y       =>      X -> Y
        -------------              --------
    So         X            =>        X
```

the conditions for adequacy are not met in the second premiss. In consequence even if we had determined that the symbolisation was actually invalid we would not have determined the invalidity of the English argument.

As with with validity we are also able to prove more general conditions under which the invalidity of an argument follows the invalidity of its symbolisation. It can be shown that [15]

a') If in a consistent set any purely positively occurring component is replaced by a statement that it logically implies the resulting set is consistent. b') If in a consistent set any purely negatively occurring component is replaced by a statement that logically implies it the resulting set is consistent.

Paraphrased to apply to arguments via the observation that an argument is invalid if and only if the set consisting of its premisses and the negation of its conclusion is consistent we have

*') Any argument obtained from an invalid argument by replacing - any purely negative occurrence of a component in the premisses by a statement that logically implies it or any purely positive occurrence in the premisses by a statement that it logically implies or any purely negative occurrence in the conclusion by a statement that it logically implies or any purely positive occurrence in the conclusion by a statement that logically implies it - will also be invalid.

When they hold these conditions enable us to determine from a knowledge of the invalidity of a symbolisation that the argument it symbolises is invalid. We should stress that the symbolisation actually be invalid and not just have an invalid form, since as already observed the former does not follow from the latter.

Our main use of these ideas in this context is to encourage caution when making the inference from invalidity in a logic-like formalism to invalidity in the argument it is taken to represent. The adequacy conditions can be used to predict and explain trouble-spots. Because it is easy to be incautious about a "No" answer we consider an elementary example from prolog.

Suppose we have a mapping T that routinely represents English sentences of the form 'All F are G' in prolog by rules of the form 'g(X) :- f(X).' Thus 'All John's children are asleep' might go into 'asleep(X) :- child_of_john(X).' Where the logically untutored would take this natural language information to warrant a "Yes" answer to the question 'Are some of John's children asleep?' prolog replies "No" to its representation '? - asleep(X),child_of_john(X).'. This elementary question-answering system is unreliable and naive users relying on it would be misled. The problem is that in contrast to sentences of the form 'Any F are G' sentences of the form 'All F are G" often convey information that logically implies but is not logically equivalent to its usual prolog representation. Where this information is represented in the database or premisses false "No" answers can occur. Analogous remarks apply to the representation of indicative conditionals.

Since these forms of representation are clearly routine and within the domain of the usual mapping T that programmers use when representing information in prolog (and sometimes program the machine to perform)) the upshot is that while as we noted above for pure prolog one can expect "Yes" answers to be reliable one cannot have the same confidence concerning "No" answers. The expressive constraints of pure prolog syntax typically restrict suspect representations to contexts where they will do no harm if the answer is "Yes", but these same contexts are potential trouble-spots when the answer is "No".

ACKNOWLEDGEMENTS

Some of the central ideas in this paper benefitted from being presented at the University of Sussex to Aaron Sloman's graduate seminar in philosophy and to the Cognitive Studies Seminar. Thanks to all concerned, especially Carl Bache, Jon Cunningham, Alan Frisch, Gerald Gazdar, John Gibson, Tom Khabaza, Jonathon Laventhol, Rudi Lutz, Chris Mellish, Allan Ramsay and Aaron Sloman.

NOTES

[1] This example is taken from Pospesel p190. A reasonable cross-section of counter-intuitive arguments can be found in Hunter (1983). [2] An argument of this kind was suggested in conversation by Jon Cunningham. [3] If this second premiss was not necessary arguments of the form 'If X then Y, If X then not-Y so not-X' could not have consistent premisses. [4] Grice (1975) is one. Jackson (1979) contributes to the Gricean programme but neither adequately refutes the arguments of Cohen (1971). [5] Discussed in Bell and Staines (1981) Ch. 2. [6] Discussed in

Staines (1981). [7] This argument shows that the number of premisses (>0) is incidental. [8] A proof can be found in Staines (1981) pp11-14 and an introductory account is given in Halpin and Girle 1981) pp186-198. [9] We omit inner parentheses hereafter where no ambiguity results. [10] Note the contrast between these and productions which require that a fact not be present in the database if they are to fire. [11] See e.g. Hammond (1983). [12] This is not true for impure prolog with negation as failure in the body of the rules. Some of the difficulties in this case stem from the fact that the body of a rule is a purely negative context and unless some strong assumptions are made failure does not imply negation. There is a good discussion in Pereira (1983). [13] Massey (1981) discusses some of them. [14] Although it may be true it does not follow that every natural language argument symbolised in this way is invalid. [15] There is a proof in the appendix to Staines (1981) .

REFERENCES

Bar-Hillel Y. (ed) Pragmatics of Natural Language, Dordrecht, Reidel 1971

Barr A. and Feigenbaum E. (Eds) The Handbook of Artificial Intelligence Vol 1 William Kaufmann Inc. 1981

Bell P.B. and Staines P.J. Reasoning and Argument in Psychology, Routledge and Kegan Paul London 1981

Cohen L.J. The Logical Particles of Natural Language, in Bar-Hillel (1971) pp50-68

Cole P. and Morgan J.L. (eds) Syntax and Semantics 3: Speech Acts Academic Press, New York 1975

Coles L.S. Techniques for Information Retrieval Using an Inferential Question-Answering System with Natural-Language Input, SRI Final Report 1972)

Clocksin W. and Mellish C., Programming in Prolog, 1981, Springer-Verlag, Berlin

Davis, R. and King, J. "An Overview of Production Systems" in Elcock and Michie 1977, pp300-332.

Elcock E. and Michie D. (eds) Machine Intelligence 8, Ellis Horwood 1977

French P., Uehling T. and Wettstein H. The Foundations of Analytic Philosophy, Midwestern Studies in Philosophy Volume VI, 1981 Grice H.P. Logic and Conversation in Cole and Morgan (1975) pp41-58

Gutenplan S. and Tamny M. Logic; a Comprehensive Introduction, Basic Books New York 1978

Halpin T. and Girle R. Deductive Logic (2nd Ed) Logiqpress, Strathpine, 1981

Hammond P. Representation of DHSS Regulations as a Logic Program, in Proceedings Expert Systems 83 Conference (Cambridge 14-16 Dec) pp225-235

Hunter, G. The Conditional, The Aristotelian Society, Supplementary Vol. LVI1 1983, pp 1-15

Jackson F. Assertion and Indicative Conditionals, Philosophical Review LXXVIII No 4 1979 pp565-589

Kalish D. and Montague R. Techniques of Formal Reasoning, Hardcourt Brace and World, 1964

Kowalski R. Logic for Problem Solving, North Holland, 1979

Hassey 6. The Fallacy Behind Fallacies, in French, Uehling and Hettstein 1981 pp489-500 •

Pereira, F. Logic for Natural Language Analysis, SRI Technical Note 275, 1983

Pospesel H. Propositional Logic; introduction* to logic, Prentice Hall New Jersey 1974.

Staines P.J. Some Formal Aspects of the Argument Symbolisation Relation Australian Logic Teacher's Journal Vol 5 No 3 1981, pp 1-15

Winston P.H. Artificial Intelligence Addison-Wesley, 1977