Research report 173

FINDING PROBLEMS IN KNOWLEDGE BASES USING MODAL LOGICS

IAN P GENT

(RR173)

In this paper I propose that it is suitable to consider some statements that an expert makes during knowledge elicitations as being statements in a modal logic. This approach gives us several advantages in finding inconsistencies between a knowledge base and an expert's intuitions of her field. I illustrate this approach by using the modal logic VC, a logic of counterfactual conditionals. In an appendix, I give brief details of theorem proving in VC.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

February 1991

Finding Problems in Knowledge Bases Using Modal Logics

Ian P. Gent

Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom

e-mail ipg@cs.warwick.ac.uk

Abstract

In this paper I propose that it is suitable to consider some statements that an expert makes during knowledge elicitation as being statements in a modal logic. This approach gives us several advantages in finding inconsistencies between a knowledge base and an expert's intuitions of her field. I illustrate this approach by using the modal logic VC, a logic of counterfactual conditionals. In an appendix, I give brief details of theorem proving in VC.

1 Introduction.

How often does it happen that a knowledge engineer, after building a knowledge base, is confronted with a statement from his[1] expert such as "*But of course, if the moon were made of green cheese, landing on it would still be possible*" only to discover that this statement is inconsistent with the knowledge base?

In this paper, I suggest that such events should be taken seriously, and I propose a way in which this can be done with logical rigour. By translating such statements into a suitable modal logic, we can analyse both the knowledge base and these statements together. This technique enables more of the expert's knowledge to be utilised, and goes beyond simply checking the consistency of knowledge bases, as for instance done by Beauvieux and Dague [1990]. Furthermore, this analysis does not rely on a human noticing inconsistencies, since theorem provers are available for modal logics.

In the examples in this paper, I will assume that an expert makes *counterfactual conditional* statements, following Bench-Capon's [1989] suggestion that counterfactual statements are suitable tools for analysing knowledge bases. A counterfactual conditional statement is one in which the antecedent to the conditional is false. The statement about the moon in the first paragraph above is a counterfactual. Unlike Bench-Capon, however, I will use Lewis's [1973] logic of counterfactuals VC.

The use of a particular counterfactual logic is not central to the idea this paper supports. In fact, even the use of counterfactual statements is not central. The crucial point is the very decision to use a logic for analysing statements an expert may make that will not fit into the logic of the knowledge base. In different applications, a temporal logic, for example, may be more suitable, for instance, if the expert is liable to make statements such as "Event A always happens before Event B". Typically I would expect a modal logic to be most suitable. Thus my paper is situated firmly in the "neat" or "logicism" camp of AI (depending on whether you prefer Alan Bundy's or Drew McDermott's term).

I urge the use of a logic in this application for various reasons. Firstly, given a logic, we can collect together the statements the expert makes and treat them as a whole. Using Bench-Capon's approach we have to treat each statement separately. The same would apply were we to use Matthew Ginsberg's [1986] approach to counterfactuals, which can only deal with single counterfactuals. Secondly, we can enforce a very clean separation between the statements of the knowledge base the engineer writes, and the supporting statements of the expert. That is, we test the knowledge base in the presence of the aggregate of the latter, and search for inconsistencies. If we fail to find any, and are happy with the knowledge base, we can discard the supporting statements before putting the knowledge base into action. Thirdly, we get the advantages that logicists are always happy to claim. That is, the use of a logic provides clearly stated semantics. This is something that Bench-Capon and Ginsberg do not give us. This then enables us to separate out the problem of writing the program to correctly deal with this semantics, and the application domain in which we use the program. Then, as in this paper, we can analyse an approach to solving a problem separately from the implementational issues. Thus we have a theory of what is going on that is not dependent on deep understanding of implementational issues. This makes falsification of such a theory much easier.

Note that in this paper I use logic while the application domain is knowledge bases, which usually use only a subset of a logic. It may seem that this means that I am using a sledgehammer to crack a nut. However, my colleague David Randell (personal communication) points out that we use logic to *model* a situation. In this case I am modelling the process of finding inconsistencies between an expert's intuitions about a

[1] For ease of writing, I decided that throughout this paper I would assume that the expert and knowledge engineer were of different sexes. The toss of a coin decided that the expert was to be female and the knowledge engineer male.

domain and a knowledge base about that domain. However, it is often possible, in particular applications, to use just some features of a logic, and thus ease computational problems. In such a case, we can build an implementation that reflects the crucial aspects of our logical model, and performs identically to the logical model on those aspects. An example of this would be the use of PROLOG instead of a full theorem prover for sets of Horn clauses. Thus in this paper, I propose an analysis of what should be done to find inconsistencies while admitting that the particular implementation I use may be inadequate in practice.

2 Assumptions and Notation

There are two primary assumptions on which this work is based.

I assume that it makes sense to divide the knowledge base into statements that are necessarily true, and other statements that are mere contingent facts. I do not wish to imply that the necessary statements have to be universally true in all imaginable situations, but merely that, for a particular application, it is sensible to regard them as unbreakable rules. This is a reasonable assumption, since if a knowledge base is released to the world for use, it is going to have to work in a wide variety of situations and so its rules should have applicability to all those situations. Equally, it is reasonable to assume that a knowledge base will also contain facts that are not universally true, but that may vary from one situation to another. For example, a knowledge base may contain the rule "If a car has four wheels, it must have a spare tyre" considered as universally true and the fact "Metros have four wheels" considered as a contingency since we do not know what Metro designers will do next year.

I also assume that there are various statements an expert may make that cannot be put into the knowledge base. Such statements might include modal, counterfactual, and temporal statements which could not go into a knowledge base using classical logic. I will call these statements the expert's "Hypothetical" statements. Further, I assume that there is a logic available for dealing with such statements, and which contains the logic of the knowledge base as a subset. This means that I also make the lesser assumption that the knowledge base uses a logic. Although this will not always be true, it is a reasonable assumption since the only logical property I will be using is inconsistency, and it is reasonable to look for inconsistency in a knowledge base even if the knowledge base does not use a full version of some logic.

Having made these assumptions, I will introduce some notation.

L will be the logic that the knowledge base is written in.

KB will be the conjunction of all the statements in a knowledge base.

KB_N will be the conjunction of all the necessary statements in a knowledge base.

KB_C will be the conjunction of all the contingent statements in a knowledge base.

M will be the modal logic that the hypothetical statements may be written in.

I assume that one of the connectives in **M** is $\Box$, where "$\Box\phi$" is read as "ϕ is necessarily true". Since KB_N is a statement of **L**, and we wish to assert that KB_N is necessarily true, the appropriate statement in **M** is $\Box KB_N$.

Hyp will be the conjunction of all the expert's hypothetical statements.

Finally, the entailment relations of **L** and **M** are $\models_{\mathbf{L}}$ and $\models_{\mathbf{M}}$ respectively.

Note that my assumption that **L** is contained in **M** means that if $\phi \models_{\mathbf{L}} \psi$ then $\phi \models_{\mathbf{M}} \psi$.

3 Methodology

My suggestion is that the knowledge engineer should keep a note of the expert's hypothetical statements, and translate them into **M** . Periodically, the engineer should test whether

$$Hyp \models_{\mathbf{M}} \neg (KB_C \ \& \ \Box KB_N)$$

If this is in fact so, then the knowledge base is inconsistent with the expert's hypothetical statements. In this case, the knowledge base needs revision. If not, then there is no evidence that the knowledge base is inconsistent with the hypothetical statements.

The reader may point out that I have said nothing about the problem of knowledge elicitation of the set of statements *Hyp* . In fact, it is clear that in general the same problems arise in trying to elicit the hypothetical statements as the knowledge base itself. However, there are some facts which alleviate this problem, and justify my implicit assumption that developing the set *Hyp* is essentially easy.

Firstly, there is no need for *Hyp* to be in any sense complete. We will never make *Hyp* available to the user, and it does not have to cover all conceivable situations. Its only use is in testing the knowledge base. Thus, the problems of deciding when a knowledge base is complete do not arise.

Secondly, the sort of statements that arise in *Hyp* are likely to be far more natural to the expert than the statements required for the knowledge base. In effect, in asking for hypothetical examples, we expect to get much closer to the expert's intuitions about her field. Furthermore, she is much more free in the statements she can make. There is no need to generalise, since particular examples are satisfactory, and we need not restrict her to a particular form of statement, such as rules.

Finally, there is one pleasant fact about the methodology I propose here. That is that the sets *Hyp* and *KB* are developed completely separately. Their only interaction is in *testing*. This means that when the testing has been completed and it has been found that *Hyp* and *KB* are consistent (i.e. it is not true that $Hyp \models_{\mathbf{M}} \neg (KB_C \ \& \ \Box KB_N))$, *KB* is ready for use as a normal knowledge base. This means that there is no need to keep track of which facts have been added to *KB* for testing and will have to be removed later, or any other such complication in writing *KB* .

4 The Counterfactual Logic VC

Bench-Capon [1989] has suggested that counterfactual statements are suitable for this application, and that this is so is reflected by the statement that opened this paper. Unlike Bench-Capon, however, I will use a *logic* of counterfactuals. In particular, I will use the logic VC developed by Lewis [1973]. This choice is partly motivated by the fact that I have a theorem prover for this logic to hand. In the rest of this section I introduce the logic VC very briefly. Formal details of VC, including details of a theorem proving method, are included in the appendix.

To simplify the presentation in this Section, I will make one or two silent assumptions that Lewis does not. This will not affect the feel of the theory, although it would make a difference in some important cases.

Lewis assumes that there exist, as well as the actual world, a set of other *possible worlds*. These other worlds are of the same kind as the actual world: he does not ascribe any special status to the actual world, except that it is the one we happen to be in.

Next, Lewis supposes that for each world i, there is a relation $\leq_i$ which applies to pairs of possible worlds. "j $<=_i$ k" is read as "world j is more similar to world i than world k is". Lewis demands that any two worlds j and k must be comparable by this relation, although distinct j and k may be indistinguishable if j $\leq_i$ k and k $\leq_i$ j. Since either j $\leq_i$ k or k $\leq_i$ j, it makes sense to define a strict version of this relation, $<_i$, by j $<_i$ k iff not(k $\leq_i$ j). In addition to demanding that any two worlds must be comparable, Lewis also demands that each relation $\leq_i$ must be transitive, and be such that i $\leq_i$ k for any k.

Given this apparatus, we can define semantically what it means for a counterfactual conditional to be true at a given world. The counterfactual conditional "if ϕ were the case then ψ would be the case" is written as "ϕ $\Box\!\!\Rightarrow$ ψ".[2]

Lewis considers ϕ $\Box\!\!\Rightarrow$ ψ to be true at a world i if ψ is true in all the nearest worlds with respect to $\leq_i$ where ϕ is true, and there is at least one world accessible from i where ϕ is true.

On introspection, this theory seems attractive. When trying to work out whether ϕ $\Box\!\!\Rightarrow$ ψ is true, I often imagine what the world would be like if ϕ were true. The problem is of course that there are any number of ways the world could be if it was different to how it is: how can you make sense of all this? Typically, I try to imagine the world with as few changes as possible to make ϕ true, and then look at ψ in that context. This seems to be close to Lewis's semantics.

Lewis also allows us to make statements about the comparative possibility of propositions. That is we can say things like "It is more likely that the moon would be made of green cheese than purple cheese". He then gives a semantics for comparative possibility similar to that for counterfactuals, and then shows that, given these semantics, the two notions can each be defined in terms of the other.

5 Example

This example illustrates the iterative development of a knowledge base, informed by the use of an expert's hypothetical statements. The first knowledge base in the example is given by Bench-Capon [1989, pp 56-57]. However, later examples diverge from his example. This is because I wish to illustrate features here that could not be reproduced in Bench-Capon's approach. Although my example starts in the same way as his, the processes involved are much more generalisable, and also capable of much greater subtlety.

One point about my knowledge base is that first order variables occur in it, even though VC is a propositional logic. Therefore, in this case the knowledge bases would have to be entered to the VC prover with all relevant individuals replacing the variables. This is not an important dodge for two reasons. First, my main point is not dependent on features of the logic I am using. Secondly, I can see no obvious problems in defining a first order analogue of VC over and above those found in other modal logics.[3]

Now for the example. In all cases, I have run the appropriate test using a theorem prover for VC that I have implemented and in each case it gave the response reported here.

Suppose that a knowledge engineer has arrived at the following knowledge base, in consultation with an expert on military history.

Knowledge Base 1

Necessities

((ruthless(C) & available(a-bomb,C)) => used(a-bomb,C))
& (commandK(C) => available(a-bomb,C))

Contingencies

commandK(mcarthur)

And at this point, the expert says "Of course, if Caesar had been in command in Korea,

---

[2] Actually, Lewis prefers another connective, "$\Box\!\!\rightarrow$". This is read in the same way as "$\Box\!\!\Rightarrow$", but the truth conditions are weaker. In this paper "$\Box\!\!\Rightarrow$" fits my purpose better.

[3] Lewis dislikes the idea of cross world identity of individuals (see Lewis [1973], §1.9), and for this reason did not give a first order presentation of VC.

he would not have used the atom bomb". This is clearly a counterfactual conditional, since in fact Caesar did not command in Korea. Thus it is not suitable for inclusion in the knowledge base, so we put it into a parallel set of hypothetical statements. The expert also informs us that Caesar was ruthless as a supporting statement. To represent this we must assert it as necessarily true, since it does not vary over the situations we are considering.

Hypothetical Set 1

commandK(caesar) □⇒ ¬ used(a-bomb,caesar)
& □ ruthless(caesar)

We now check whether

$$Hyp \models_M \neg(KB_C \ \& \ \Box KB_N \ )$$

In fact, it turns out that it does. This is because *Hyp* entails the existence of a world in which commandK(caesar) is true and used(a-bomb,caesar) is false. But KB_N & ruthless(caesar) must hold at that world, and together these entail used(a-bomb,caesar).

The engineer concludes that his knowledge base is flawed. The knowledge engineer has to revise his knowledge base. I have no suggestions for how he may do this, but let us suppose that he comes up with the following attempt.

Knowledge Base 1 *revised*

Necessities
((ruthless(C) & available(a-bomb,C) & ¬careful(C)) => used(a-bomb,C))
& ((ruthless(C) & available(a-bomb,C) & careful(C) & understood(a-bomb,C))
=> used(a-bomb,C))
& (commandK(C) => available(a-bomb,C))

Contingencies
commandK(mcarthur)

Running the VC theorem prover with the same hypothetical set but with the new knowledge base, the engineer discovers that *Hyp* and $\Box KB_N$ are consistent. Therefore he concludes that the knowledge base is consistent with the expert's other statements about the subject.

As the engineer discusses atom bombs further with the expert, she may say "It's not true that if Einstein had been ruthless and in command in Korea, he would have used the bomb". The supporting information is that (across all situations we are considering) Einstein would certainly have understood the atom bomb. This is translated in VC and added to the first hypothetical set to produce:

Hypothetical Set 2

((commandK(caesar) □⇒ ¬ used(a-bomb,caesar)))
& (□ ruthless(caesar))
& (¬ ((ruthless(einstein) & commandK(einstein)) □⇒ used(a-bomb,einstein)))
& (□ understood(a-bomb,einstein))

Note that the second counterfactual statement above is of a different form to the first (in that one asserts the truth of a counterfactual, and the other the falsity). There is no problem about handling this situation in VC, or in other modal logics, and hence no need to get the expert to rephrase her statement.

Now, however, we find that *Hyp* is inconsistent with KB_N. This is because the revised knowledge base entails that a ruthless Einstein would have used the bomb, which is contradicted by the hypothetical statement of the expert. However, the proof that this is so is of some subtlety and could easily be missed by the knowledge engineer. Had Einstein been in command in Korea, as we are forced to consider by the connective "□⇒", we could assert that either he would have been careful or not. In the former case, the second rule in the knowledge base asserts that he would have used the atom bomb. In the latter case, it is the first rule that asserts he would have used the bomb. The nested use of the law of the excluded middle is the sort of argument that a knowledge engineer might miss. The use of a theorem prover assures us that lapses like that will not occur.

Let us assume that the knowledge engineer revises the knowledge base again.

Knowledge Base 2

Necessities

((ruthless(C) & available(a-bomb,C) & ¬scientist(C)) => used(a-bomb,C))
& ((ruthless(C) & available(a-bomb,C) & scientist(C)) => ¬used(a-bomb,C))
& (commandK(C) => available(a-bomb,C))

Contingencies

commandK(mcarthur)

This change in the knowledge base requires a change in the hypothetical set, to take account of the change from the concept of "understanding" to "scientist". Of course Einstein is a scientist.

Hypothetical Set 2 *revised*

((commandK(caesar) □⇒ ¬ used(a-bomb,caesar)))
& (□ ruthless(caesar))
& (¬ ((ruthless(einstein) & commandK(einstein)) □⇒ used(a-bomb,einstein)))
& (□ scientist(einstein))

Now the knowledge base is consistent in VC with the hypothetical set.

At this point, the expert may say, "But if Caesar had been in command in Korea, he would not have been a scientist". Translating this into VC and adding it to the hypothetical set we get

Hypothetical Set 3

(commandK(caesar) □⇒ ¬ used(a-bomb,caesar))
& (□ ruthless(caesar))
& (¬ ((ruthless(einstein) & commandK(einstein)) □⇒ used(a-bomb,einstein)))
& (□ scientist(a-bomb,einstein))
& (commandK(caesar) □⇒ ¬scientist(caesar))

If we check now the Hypothetical Set 3 against the Knowledge Base 2, we discover that they are inconsistent again. This is because the statements about Caesar together imply that in the nearest world where Caesar was in command in Korea, he was ruthless, did not use the bomb, and was not a scientist. This contradicts the knowledge base, which asserts that a ruthless non-scientist in command in Korea would have used the bomb.

This inconsistency has only been discovered because we preserved all the statements of the expert and then tested them all together against the knowledge base. This is a direct result of the principled methodology we have been using. Using a less principled method, perhaps we would have tested statements together, only to discard the statement about Caesar when we added the statement about Einstein. Also, we have benefited from using the full power of a logic. If we had only acted by adding statements about Caesar, and then Einstein to the knowledge base and then testing for consistency (and this is effectively what Bench-Capon did in his paper) we may well have only tested one statement at a time.

Knowledge Base 3

Necessities

((ruthless(C) & available(a-bomb,C) & ¬scientist(C) & ¬careful(C))
=> used(a-bomb,C))
& ((ruthless(C) & available(a-bomb,C) & scientist(C)) => ¬used(a-bomb,C))
& (commandK(C) => available(a-bomb,C))

Contingencies

commandK(mcarthur)

Now the engineer has reestablished consistency between the knowledge base and the hypothetical statements. The game goes on.

6 Concluding Remarks On Computational Issues

The approach outlined above suffers from one major disadvantage that may not be ignored. This is that theorem proving in these modal logics is usually extremely expensive computationally compared to typical inference schemes in knowledge based systems. For example, theorem proving in the counteractual logic VC used in this paper is PSPACE-complete[4] (see the Appendix). Of course, first order versions of modal logics are undecidable. These problems cannot be avoided using even the best theorem provers, since they are inherent properties of the logics.

Of course, slow computation is not the most important factor here, since theorem proving is only done during knowledge elicitation, and not when the knowledge base is put to use. Also, much work is going on in finding efficient theorem provers for modal logics, see for instance work by Wallen [1987, 1989] and Ohlbach [1988]. However, the size of a realistic knowledge base and the truly awful computational property of undecidability are likely to combine to make even the cpu time required during knowledge elicitation unacceptable.

It is natural then to ask what is the point of this work?

This work attempts to further the cause of logicism in AI. It does this by showing that a logical analysis can be used to help check knowledge bases. By using a logical analysis, we make it quite clear what assumptions we are making, and then we need nothing else. In this case, I only used the assumptions of §2. Having done this, we benefit from the logical analysis because we now have a *theory* of the problem we are attacking. Once this theory is accepted, it then remains a problem to put into practice computationally. However, we have separated out the theory from the implementation.

[4] A problem is PSPACE-complete if some algorithm can solve it using only a polynomial amount of memory space, and if it can be used to solve every other PSPACE problem using a polynomial time translation.

It is interesting to compare this approach with the one taken by Bench-Capon [1989]. He suggested the idea of using counterfactuals as an aid in knowledge elicitation. However, he was working solely in Prolog, and thus the form of his assertions were restricted to Horn clauses, and he had to use negation as failure instead of real negation as I have used here. Also, his approach to testing the counterfactual was to add the antecedent (for example "commandK(caesar)") to the knowledge base and test the result to see if "used(a-bomb,caesar)" was a consequence.

The contrast with my approach is startling: I used the full logic instead of Prolog's restriction of it; I was not restricted to his particularly simple and inextensible analysis of counterfactuals; and finally I was not even restricted to counterfactuals at all!

Even this one-sidedness is not the crucial point I wish to make. The point is that the conceptual analysis I have developed in this paper applies equally even if we do accept Prolog's logic and Bench-Capon's analysis of counterfactuals. That is, the analysis developed here tells us what to look for if we have a knowledge base written in some logic and a way of reasoning about modal statements that fall outside that logic. In this case, we have to consider how we would analyse a set of counterfactual statements using Bench-Capon's analysis. This is not a trivial matter, since he only tells us how to analyse one at a time. The problem of implementing our conceptual analysis has raised an issue that was not obvious in the original paper.

This is not to say that analyses like Bench-Capon's are worthless. If, in a particular application, they retain the salient features of a more complex analysis of counterfactuals, but are more efficient computationally, then they are worthwhile. The effort involved in implementing correctly the tests required by this paper may also be worthwhile. The point I wish to stress is that no changes are required to the analysis presented in this paper.

There is nothing special about this example of a special case. Given any restriction of logic used by a knowledge base, and some way of deciding statements that are outside that logic in some way, this theory suggests a methodology that the knowledge engineer should go through.

The use of VC in this paper gave me several advantages, in that I never had to discard the expert's statements and that different statements in the hypothetical set interacted without me having to worry about it. This does not mean that it always be best to use computationally expensive logics. As David Randell pointed out to me (personal communication), in particular situations we should look for the theories that fit the situation and are cheapest computationally.

Appendix: Formal Details of VC

Formal Semantics of VC

The language of VC contains a set of propositional constants A, B, . . . , standard propositional connectives $\wedge$, $\vee$, $\neg$, $\supset$, (but I will give formal details only for $\neg$ and $\supset$) and the extra connectives $\Box\!\!\Rightarrow$ and $\leq$. $\leq$ is another connective in VC additional to $\Box\!\!\Rightarrow$. It should not be confused with the semantic relation $\leq$ which will be introduced in a moment. "$A \leq B$" is read as "It is at least as likely that A would be true as that B would be true". The connective $\Box\!\!\Rightarrow$ is used for counterfactual implication; "$A \Box\!\!\Rightarrow B$" is read as "If A were the case, then B would be the case". However, we need only consider the connective $\leq$ since $\Box\!\!\Rightarrow$ can be defined in terms of $\leq$ by $A \Box\!\!\Rightarrow B \equiv \neg\,(\,(A \wedge \neg B) \leq (A \wedge B)\,)$.

The following definition gives the semantics of VC.

A *model for VC* is a quadruple (I , R , $\leq$, [ ]) which satisfies:

(1) I is a nonempty set of possible worlds.

(2) R is a binary relation on I, representing the mutual accessibility relation of possible worlds.

(3) $\leq$ is a three place relation on I, s.t. for each $i \in I$ there is a binary relation $\leq_i$ on I. Furthermore, each $\leq_i$ must be transitive and connected on $\{j \mid j \in I \text{ and } iRj\}$. (The latter requirement is that if iRj and iRk then either $j \leq_i k$ or $k \leq_i j$ or both must be true).

(4) [ ] assigns to each formula A of VC a subset [A] of I, representing the set of worlds where A is true. [ ] must satisfy the following requirements, for each element i of I:

(4.1) $i \in [\neg A]$ if and only if $i \notin [A]$.

(4.2) $i \in [A \supset B]$ if and only if $i \notin [A]$ or $i \in [B]$.

(4.3) $i \in [A \leq B]$ if and only if for all $j \in [B]$ s.t. iRj, there is some $k \in [A]$ s.t. $k \leq_i j$.

(5) *(The Centering Assumption)*
R is reflexive on I; and if iRj and $i \neq j$ then $\neg\, j \leq_i i$, (and so by the connectivity of $\leq_i$ and reflexivity of R, and in an obvious notation, $i <_i j$).

A formula A is a *theorem of VC*, written "$\vdash_{VC}$ A", if and only if, in every model $(I, R, \leq, [\,])$ for VC, $[A] = I$.

Theorem Proving in VC

This section is only the briefest of summaries. For full details see Gent [1990].

De Swart [1983] gave a sequent system for VC, but it is incorrect. The following formula is a theorem of VC, but his method fails to prove it so.

$$((A \leq C) \wedge (C \leq D) \wedge (D \leq (\neg D \wedge B))) \supset (A \leq B)$$

The decision problem for VC is PSPACE-complete. This can be shown by first reducing the decision problem for the modal logic T to that of VC. T is PSPACE-hard, as shown by Ladner [1977]. That VC is PSPACE-easy can be shown by proving that the following method can be implemented in polynomial space.

I now give a very brief description of a sequent based proof system for VC, which I have implemented in PROLOG. To prove a theorem one must find a tree built according to the rules given below in which each leaf sequent contains both a formula and its negation, with the root node being the negation of the theorem. The sequent rules are as follows.

T¬	S, T¬B S, FB	F¬	S, F¬B S, TB
T⊃	S, TB⊃C S, FB ∣ S, TC	F⊃	S, FB⊃C S, TB, FC
T≤	S, TB≤C S, TB≤C, TB ∣ S, TB≤C, FC	F≤	S, FB≤C S, FB≤C, FB

There is one additional rule that is considerably more complicated. Its general name is F≤(m,n). It applies to a set of m formulas of the form FA≤D and n formulas of the form TU≤V. It is only applicable if $m \geq 1$, but n may be 0. The definition of F≤(m,n) is given below. Note that in the rule F≤(m,n), the sequent S in the upper sequent does not appear in the derived sequents. This is because the derived sequents can be seen as referring to different possible worlds and so many statements about the original world become irrelevant.

$$F{\le}(m,n) \quad \frac{S,\ FA_1{\le}D_1, \ldots, FA_m{\le}D_m, TU_1{\le}V_1, \ldots, TU_n{\le}V_n}{S_1 \mid S_2 \mid \cdots \mid S_m \mid (*)}$$

where $S_i = \{FA_1, \ldots, FA_m, TD_i, FV_1, \ldots, FV_n\}$ for $1 \le i \le m$

and where (*) is the following special condition, which only applies if $n \ge 1$.

(*) There is a sequence $i_1, i_2, \ldots, i_n$ which is a permutation of $1,2,\ldots,n$ and is such that each of the following sequents is derivable.

$\{FA_1, \ldots, FA_m, TU_{i_1}\}$

$\{FA_1, \ldots, FA_m, TU_{i_2}, FV_{i_1}\}$

$\{FA_1, \ldots, FA_m, TU_{i_3}, FV_{i_1}, FV_{i_2}\}$

...

$\{FA_1, \ldots, FA_m, TU_{i_n}, FV_{i_1}, FV_{i_2}, \ldots, FV_{i_{n-1}}\}$

Acknowledgements

I wish to thank Tony Cohn, Kathy Courtney, Rajeev Goré, Felix Hovsepian, David Randell, Guy Saward, and the Science and Engineering Research Council, for their help in producing this paper.

References

Alain Beauvieux, and Philippe Dague, "A general consistency (checking and restoring) engine for knowledge bases", Proceedings of the European Conference on Artificial Intelligence, 1990, pp 77-82.

T J M Bench-Capon, "Representing Counterfactual Conditionals", pp. 51-60 in *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour*, ed. A G Cohn, Pitman / Morgan Kaufmann, 1989.

de Swart, H.C.M., A Gentzen- Or Beth-Type System, a Practical Decision Procedure and a Constructive Completeness Proof for the Counterfactual Logics VC and VCS, *J. Symbolic Logic* **48** (1983), 1-20.

Gent, I.P., *A Sequent System for Lewis's Counterfactual Logic VC*, Research Report number 168, Department of Computer Science, University of Warwick, 1990.

Matthew L Ginsberg, "Counterfactuals", *Artificial Intelligence* **30** (1986) 35-79.

Richard E Ladner, "The computational complexity of provability in systems of modal propositional logic", SIAM J Comput **6** (1977) 467-480

David Lewis, *Counterfactuals*, Basil Blackwell, Oxford, 1973.

H J Ohlbach, "A resolution calculus for modal logics", CADE 9, Springer Verlag, pp 500-516, 1988.

Lincoln A Wallen, "Matrix Proof Methods for Modal Logics", IJCAI 1987, Morgan Kaufmann, pp 917-923.

Lincoln A Wallen, *Automated Deduction in Non-Classical Logics*, MIT Press, Cambridge, Mass., 1989.