PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1984,
edited by ICOT. © ICOT, 1984

# INCIDENCE CALCULUS:

# A MECHANISM FOR PROBABILISTIC REASONING

Alan Bundy

Department of Artificial Intelligence,
University of Edinburgh,
Hope Park Square,
Edinburgh, EH8 9NW, Scotland.

## Abstract

Mechanisms for the automation of uncertainty are required for expert systems. Sometimes these mechanisms need to obey the properties of probabilistic reasoning. We argue that a purely numeric mechanism, like those proposed so far, could not do so. We propose an alternative mechanism, **Incidence Calculus**, which is based on a representation of uncertainty using sets of incidents, which might represent situations, models or possible worlds. Incidence Calculus does obey the properties of probabilistic reasoning.

## 1 INTRODUCTION

Several mechanisms have been suggested for the automation of reasoning with uncertainty, e.g. Fuzzy Logic, [Zadeh 81], Shafer-Dempster theory, [Lowrance & Garvey 82], and the mechanisms proposed in MYCIN, [Shortliffe 76] and PROSPECTOR, [Duda et at 78]. Most of these mechanisms involve assigning numbers to axioms (e.g the facts and rules of an expert system), and assigning arithmetic functions to the logical connectives and rules of inference, so that new numbers can be calculated for the theorems that are derived from the axioms (e.g. the diagnoses of an expert system). We will call such mechanisms, **purely numeric**.

In some applications it is important to be able to assign *meaning* to the numbers so obtained, rather than use them merely to rank order some options. For instance, in medical diagnosis a user sometimes needs to be able to distinguish the situations where a diagnosis is very probably correct from the situation where it is just the best of an improbable batch. In the first case a surgeon might be prepared to perform a dangerous operation. In the second s/he might want to call for more tests and a re-diagnosis. In these situations we would like the numbers to represent probabilities.

Unfortunately, we will see that a purely numeric mechanism *cannot* capture the properties of probabilistic reasoning. We will propose a mechanism, **Incidence Calculus**, based on assigning and manipulating sets, which *does* capture the properties of probabilistic reasoning.

## 2 PROBABILISTIC REASONING

What properties must probabilistic reasoning obey? These can be obtained from any textbook on mathematical probability, e.g. [Freund 72]. Adapting these properties to the needs of a logical calculus gives the set of equations given below.

The probability of some formula being true is a real number, between 0 and 1. We will use upper case letters from the beginning of the alphabet to denote formulae, e.g. A, B, C, etc and write p(A) for the probability of A, etc. The probability of a true formula is 1, and the probability of a false formulae is 0. That is,

$$p(t) = 1 \qquad (i)$$
and $$p(f) = 0, \qquad (ii)$$

where t represents the truth value, true, and f represents the truth value, false. Values intermediate between 0 and 1 correspond to degrees of probability between these extremes.

The following arithmetic formulae are then assigned to the propositional connectives.

$$p(\tilde{}A) = 1 - p(A) \qquad (iii)$$

$$p(A \,\&\, B) = p(A).p(B) \qquad (iv)$$
provided A and B are independent

$$p(A \vee B) = p(A) + p(B) - p(A \,\&\, B) \qquad (v)$$

As we will see, the condition attached to equation (iv) is very important. It means that B is no more or less likely given A than in general, and vice versa. It is this condition that will prevent numerical methods of representing probabilistic reasoning from succeeding, because the independence of two formulae cannot be coded in their numerical values.

## 3 THE LIMITATIONS OF A PURELY NUMERIC MECHANISM

If we ignore the independence condition on equation (iv) then we get a contradictory calculus. The contradiction can be derived by applying the rules, unconditionally, to two dependent formulae, for instance A and ¬A. Suppose, for the sake of definiteness, that p(A) = .75. Using the equations of the last section we have the following derivation.

$p(\neg A) = 1 - p(A) = .25$ (by (iii))

$p(A \,\&\, \neg A) = p(A).p(\neg A) = .1875$ (by (v))

Similarly,
$p(A \vee \neg A)$
$= p(A) + p(\neg A) - p(A \,\&\, \neg A)$ (by (iv))
$= .8125$

However, $p(A \,\&\, \neg A) = p(f) = 0$ (by (ii))

and $p(A \vee \neg A) = p(t) = 1$ (by (i))

but $.1875 \neq 0$ and $.8125 \neq 1$. Contradiction!

The contradiction cannot be avoided by modifying the arithmetic functions associated with the connectives. We would have to modify the equations (v) and (iii) so that they gave value 0 when calculating the probability of a formula of the form A & ¬B, where the probability of both A and B is .75. But not all such formulae are false.

We need to design a mechanism which can take into account the degree of dependence of formulae when calculating their probabilities. In probability theory the **correlation**, c(A,B), between two formulae, A and B, is used to measure their degree of dependence. It varies between the values -1 to 1. c(A,B)=0 means A and B are independent, c(A,A)=1 and c(A,¬A)=-1. The correlation is so defined that:

$$p(A \,\&\, B) = p(A).p(B) + c(A,B).\sqrt{p(A).p(\neg A).p(B).p(\neg B)} \qquad \text{(vi)}$$

This formula provides an unconditional alternative to formula (iv), i.e. one without the condition that the conjuncts be independent. However, it does assume knowledge of both the probabilities *and* the correlations of the conjuncts. This assumption would not be unreasonable if we had a *correlation calculus* which provided formulae for calculating the correlations of complex formulae from their subformulae, i.e. if we had formulae which enabled the calculation of c(A&B,C), c(AvB,C), c(¬A,C) and c(A->B,C) purely from p(A), p(B), p(C), c(A,B), c(A,C) and c(B,C).

We will see in section 6 that it is not possible to provide such a calculus. So in order to use formula (vi) it is necessary to be provided with the correlations of all the infinitely many, possible pairs of formulae. This means that using correlations is not a feasible solution to the problems of dependency in probability theory.

## 4 A SET THEORETIC MECHANISM

To design a mechanism which *can* deal with this problem we need to go back to the set-theoretic roots of probability theory. The probability of a formula is based on a disjoint set of situations, Tarskian interpretations or possible worlds, [Freund 72], which we will call **incidents**. **w** will represent the set of all incidents in which the formulae of the theory are to be evaluated.

Non-trivial theories often have an infinite number of possible interpretations. For computational reasons we will usually require w to be finite. Thus each incident must sometimes stand for an infinite set of interpretations by singling out some distinct properties of this set.* Let i(A) be the subset of w, containing all those incidents in which formula A is true. We will call i(A), the **incidence**** of A. The dependence or independence of two formulae is coded in the amount of intersection between their incidences. The amount of intersection of two independent formulae is no more or less than you would expect from a random assignment of the elements of their incidences.

Incidence Calculus can be added to an existing logic, e.g. Propositional or Predicate Logic, as its semantic interpretation. For instance, it provides an alternative to the truth tables of Propositional Logic, where i(A) is the alternative to the truth value of the formula A. The normal, Tarskian semantics may still be useful to settle the meaning of a formula within an incident, e.g. if that incident is a Tarskian interpretation.

For the rest of this paper we will assume the underlying logic to be Predicate Logic with formulae in fully-Skolemized form. The fol-

**incident*: a distinct or definite occurrence; event. - Collins English Dictionary.

***incidence*: degree, extent or frequency of occurrence; amount. - Collins English Dictionary.

lowing equations give a semantics to the truth functional connectives, constants and free variables of Predicate (and Propositional) Logic to give **Predicate (Propositional) Incidence Logic**.

$$i(t) = w \tag{vii}$$

$$i(f) = \{\} \tag{viii}$$

$$i(\neg A) = w \setminus i(A) \tag{ix}$$

$$i(A \mathbin{\&} B) = i(A) \cap i(B) \tag{x}$$

$$i(A \vee B) = i(A) \cup i(B) \tag{xi}$$

$$i(A(X)) \subseteq i(A(T)) \tag{xii}$$

where $A(T)$ is the formula formed by substituting term $T$ for every occurrence of $X$ in $A(X)$.

Note that there is no independence condition on equation (x). Note also that since the formula are in Skolem normal form, it is not necessary to give equations for the quantifiers.

If $I$ is an incident, let $p(I)$ be the probability of $I$ occurring. If $S$ is a set of incidents, let $wp(S)$ be the sum of the probabilities of the incidents in S, i.e.

$$wp(S) = \Sigma_{I \in S}\ p(I)$$

$wp(S)$ is called the **weighted probability** of S. For computational reasons we will usually use finite S, but the theory does not require S to be finite or even discrete.

Since the incidents of w are disjoint,

$$wp(w) = 1$$

If A is a formula, let $p(A)$ be the probability of A occurring. We define

$$p(A) = wp(i(A))$$

From these definitions it is easy to derive the probability equations of section 2.

$$p(t) = wp(i(t)) = wp(w) = 1$$

$$p(f) = wp(i(f)) = wp(\{\}) = 0$$

$$\begin{aligned} p(\neg A) &= wp(i(\neg A)) = wp(w \setminus i(A)) \\ &= [wp(w) - wp(i(A))] \\ &= 1 - wp(i(A)) = 1 - p(A) \end{aligned}$$

If A and B are independent then B is true just as frequently if A is true as it is in general. This can be expressed mathematically as:

$$wp(i(A\&B))/wp(i(A)) = wp(i(B))$$

$$\begin{aligned} \text{hence, } p(A \mathbin{\&} B) &= wp(i(A\&B)) \\ &= [wp(i(A)).wp(i(B))] \\ &= p(A).p(B) \end{aligned}$$

If the weighted probability of incidents in which A is true is added to the weighted probability of incidents in which B is true then the weighted probability of incidents in which both are true are counted twice. This can be expressed mathematically as:

$$wp(i(A)) + wp(i(B)) = wp(i(A \vee B)) + wp(i(A \mathbin{\&} B))$$

hence,

$$\begin{aligned} &p(A \vee B) \\ &= wp(i(A \vee B)) \\ &= [wp(i(A)) + wp(i(B)) - wp(i(A \mathbin{\&} B))] \\ &= p(A) + p(B) - p(A \mathbin{\&} B) \end{aligned}$$

It follows that we can represent the probability of a formula, A, implicitly, by associating its incidence, $i(A)$, with it. If we need to know the probability we can calculate $wp(i(A))$.

## 5 SOME IMPLEMENTATION SUGGESTIONS

One of the advantages of a purely numeric mechanism for uncertainty is that computers are particularly efficient at representing and manipulating numbers. They are not so efficient at representing and manipulating sets.

However, Incidence Calculus can be implemented reasonably efficiently by representing the incidences of formulae as bit strings and manipulating them with logical operations. Each incidence can be represented by a bit string of a fixed length, say 100 bits, each bit corresponding to an element of w. The longer the string, the greater the accuracy, but the greater the cost in terms of space and time. Each bit in a string is 1 or 0 according to whether the element it corresponds to is or is not in the incidence being represented.

The incidence of A & B can then be calculated by taking the logical *and* of the incidences of A and B; the incidence of A v B can be calculated by taking the logical *or* of the incidences of A and B; and the incidence of ¬A can be calculated by taking the logical *not* of the incidence of A.

To simplify the calculation of probabilities, the incidents of w can be taken as equiprobable, then:

let n(S) be the number of elements in set S

since the incidents are equi-probable,
for each incident i, p(i) = 1/n(w),

hence, for each subset, S, of w,
wp(S) = n(S)/n(w)

so, for each formula, A,
p(A) = n(i(A))/n(w) (xiii)

We can now redo the calculations of section 3, but using incidences rather than probabilities. Let w = {0,1,....,99}, which might be internally represented by a bit string, as described above. Using a w of size 100 will enable us to calculate probabilities to 2 decimal places.

Suppose A is a formula with probability .75. We will assign to A the incidence {0,1,....,74}. Now using the incidence equations of section 4:

i(¯A) = {75,76...,99} (by (ix))

i(A & ¯A) = {} (by (x))

i(A v ¯A)
= {0,1,....,74,75,....,99} (by (xi))

hence, p(A & ¯A) = 0 (by (xiii))
and p(A v¯A) = 1 (by (xiii))

which is as desired.

However, if B is a formula, independent of A, with probability .25, p(A & B) is different from p(A & ¯A). Suppose we assign to B the incidence {0,4,8,....,96}, then:

i(A & B) = {0,4,8,....,72} (by (x))
hence, p(A & B) = .19 (by (xiii))

i(A v B)
= {0,1,2,....,74,76,...96} (by (xi))
hence, p(A v B) = .81 (by (xiii))

which is correct to 2 decimal places as desired.

It is interesting to compare the implementation suggestion above with the **Probabilistic Logic** of Nilsson, [Nilsson 84]. In Probabilistic Logic incidents are partially-specified Herbrand Models, i.e. they are vectors of truth assignments to a finite sequence of formulae. Thus the incidence of each formula consists of those incidents with the value 'true' in the vector slot corresponding to that formula. The following calculation is used to find the probability of a new formula, given the probabilities of some old formulae.

(a) Construct all the incidents for the set of old and new formulae.

(b) Make an assignment of probabilities to each incident which will yield the given probabilities of the old formulae.

(c) Use this assignment to calculate the probability of the new formulae.

Step (b) above is computational impractical in general. Nilsson suggests various ways to circumvent it in special cases. Our implementation suggestion avoids the problem because our incidents are essentially meaningless, rather than being Herbrand Models, as Nilsson's are. The probability of each of our incidents is set to 1/n(w), and hence an impractical calculation is avoided. Instead of fixing the probabilities of the formulae by adjusting the probability of the incidents, it is done by adjusting size of their incidences, and this is a much less expensive calculation.

If the domain supports a more meaningful representation of incidences, then these can be used instead of bit strings. For instance, in a domain involving time, it might be possible to specify a number of hypothetical futures, e.g. it rains, it snows, there is sunshine, etc. It may then be possible to assign incidences to axioms in a principled manner. If these hypothetical futures are not equi-probable, then we must assign a probability to each of them to enable wp(i) to be calculated for each incidence. When the incidences are meaningful to the user we might expect him/her to input them direct, rather than use probabilities. In this case we may not be interested in calculating probabilities at all.

I have not implemented any of these mechanisms.

## 6 THE IMPOSSIBILITY OF A CORRELATION CALCULUS

We are now in a position to redeem the promise of section 3 to show that a correlation calculus is impossible. In particular, we will show that it is not possible to give a formulae which enabled calculatation of c(A&B,C) purely from p(A), p(B), p(C), c(A,B), c(A,C) and c(B,C).

To do this we need only exhibit two situations in each of which the values of p(A), p(B), p(C), c(A,B), c(A,C) and c(B,C) are identical, but where c(A&B,C) has different values. Such a pair of situations is

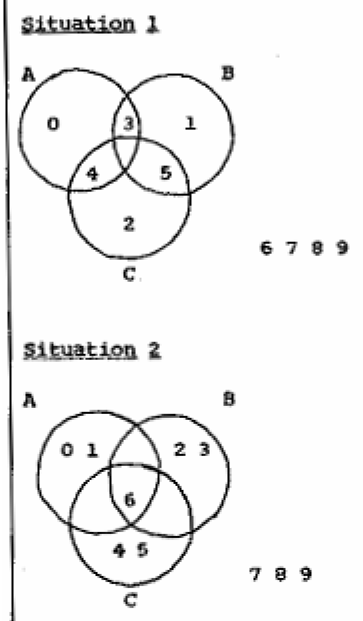

**Figure 1:** The Impossibility of a Correlation Calculus

exhibited in the Venn diagrams of figure 1. In these situations w={0,1,2,3,4,5,6,7,8,9} and each of these incidents is equi-probable. The incidences of A, B, C, A&B, etc are assigned as in the diagram. e.g. i(A)={0,3,4} and i(A&B)={3} in situation 1. From these assignments we can use formulae (xiii) and (vi) to calculate that:

$p(A) = p(B) = p(C) = 0.3$
in both situations and that

$p(A\&B) = p(B\&C) = p(A\&C) = 0.1$
in both situations.

Hence, $c(A,B) = c(B,C) = c(A,C)$
$= (0.1 - 0.3x0.3) / \sqrt{0.3x0.7x0.3x0.7}$
$= 0.047619$ in both situations.

But in situation 1 $p(A\&B\&C) = 0$, so
$c(A\&B,C) = (0 - 0.1x0.3) / \sqrt{0.1x0.9x0.3x0.7}$
$= -0.21822$

and in situation 2 $p(A\&B\&C) = 0.1$, so
$c(A\&B,C)$
$= (0.1 - 0.1x0.3) / \sqrt{0.1x0.9x0.3x0.7}$
$= -0.50918$

Considered study of this example should convince you that having different values of c(A&B,C) for the same values of p(A), p(B), p(C), c(A,B), c(A,C) and c(B,C) is not an exceptional situation, but rather the norm. So we cannot hope for a rough correlation calculus which would merely work most of the time.

Neither can we hope for a calculus based on a modified form of correlation. To be useful any such function, d(X,Y), would have to enable p(X&Y) to be calculated purely from itself and p(X) and p(Y), as in formula (vi). Thus d(X,Y) would be expressible purely in terms of p(X), p(Y) and p(X&Y). Hence, d(X&Y,Z) would be expressible purely in terms of p(X&Y), p(Z) and p(X&Y&Z). The counter-example given in figure 1 makes the values of p(A), p(B), p(C), p(A&B), p(B&C) and p(A&C) identical in the two situations, but makes the values of p(A&B&C) different. Thus d(A,B), d(B,C) and d(A,C) would be the same and d(A&B,C) would be different, however d(X,Y) was defined.

## 7 RULES OF INFERENCE

The equations of Incidence Calculus associate set theoretic algorithms with each logical connective, as an alternative to the truth tables. But, if Incidence Calculus is to be used in automatic inference, it is also necessary to associate algorithms with the rules of inference, e.g. if modus ponens is used it will be necessary to calculate i(B) from i(A) and i(A->B).

Unfortunately, there is a difficulty. In general, if H |- C is a rule of inference of a logical system then all we can infer is that $i(H) \subseteq i(C)$; if H is true in some incident then C will be true in that incident. However, C may also be true in incidents in which H is false, and so we cannot conclude that $i(H) = i(C)$. Thus, given the incidence of the axioms of a theory, we can only calculate a lower bound on the incidents of the theorems.

Each derivation of a theorem will give a lower bound on its incidence. If several different derivations give several different lower bounds then we can calculate a new lower bound by taking the union of the derived ones. This is legitimised by the set theoretic theorem,

$$L_1 \subseteq I \ \& \ L_2 \subseteq I \rightarrow L_1 \cup L_2 \subseteq I$$

where I is the incidence and $L_1$ and $L_2$ two lower bounds.

The maintenance of a lower bound of the true incidence is in the same spirit that MYCIN amalgamates the certainty factors calculated from different derivations of the same conclusion, except that the MYCIN amalgamation algorithm is ad-hoc whereas ours is justified by set theory.

In Shafer-Dempster theory both a lower and an upper bound are maintained, [Lowrance & Garvey 82]. The lower bound, Spt(A), represents the degree to which the evidence supports A: the upper bound, Pls(A), represents the degree to which the evidence fails to refute A. An upper bound could be provided in Incidence Calculus by calculating the complement in w of the lower bound of the negation.

## 8 MAINTAINING CONSISTENCY

If the user of an expert system based on Incidence Calculus is able to assign incidences in an uncontrolled manner, then it is possible to make an inconsistent assignment. For instance, it follows from the equations of Incidence Calculus that:

$$w \setminus i(A) \subseteq i(A\text{->}B)$$

so A->B cannot be assigned an incidence independently of A. Suppose that w = {a, b, c} and the user assigns i(A) = {a} and i(A->B) = {a, b}, then:

$$\begin{aligned}\{a\} = i(A) &\supseteq i(A) \cap i(\neg B)\\ &= i(A\&\neg B)\\ &= i(\neg(A\text{->}B))\\ &= w \setminus i(A\text{->}B)\\ &= \{a, b, c\} \setminus \{a, b\}\\ &= \{c\}\end{aligned}$$

so, $\{a\} \supseteq \{c\}$, which is a contradiction.

This possibility of building an inconsistent theory is true of any theory, but is particularly easy to do unintentionally in Incidence Calculus.

In Propositional Incidence Calculus, if w is finite, some of these inconsistencies can be detected by a terminating algorithm, namely:

**Definition 1:** Inconsistency Detection Algorithm:

Set up a directed graph, called the **dependency graph**, in which nodes are labelled by propositional formulae and each formula is connected by arcs to its immediate subformulae. Associate with each node the values of sup(A) and inf(A), where A is the formula labelling the node. Initialize these values as follows:

```
For each axiom, A:
 Let sup(A) := i(A)
 Let inf(A) := i(A)
 For each proper subformula,
  B, of A for which sup(B)
  and inf(B) have not already
  been assigned:
   Let sup(B) := w
   Let inf(B) := {}
```

Propagate the sup and inf values around the dependency graph, according to the following rules, until no further changes can be made.

```
For each subgraph connecting A and ~A:
 Let sup(A) := (w \ inf(~A)) ∩ sup(A)
 Let inf(A) := (w \ sup(~A)) ∪ inf(A)
 Let sup(~A) := (w \ inf(A)) ∩ sup(~A)
 Let inf(~A) := (w \ sup(A)) ∪ inf(~A)

For each subgraph connecting A, B and A&B:
 Let sup(A) := [sup(A&B) ∪ w\inf(B)] ∩ sup(A)
 Let inf(A) := inf(A & B) ∪ inf(A)
 Let sup(B) := [sup(A&B) ∪ w\inf(A)] ∩ sup(B)
 Let inf(B) := inf(A & B) ∪ inf(B)
 Let sup(A&B) := sup(A) ∩ sup(B) ∩ sup(A&B)
 Let inf(A&B) := [inf(A) ∩ inf(B)] ∪ inf(A&B)

For each subgraph connecting A, B and A v B:
 Let sup(A) := sup(A v B) ∩ sup(A)
 Let inf(A) := [inf(A v B) ∩ w\sup(B)]
               ∪ inf(A)
 Let sup(B) := sup(A v B) ∩ sup(B)
 Let inf(B) := [inf(A v B) ∩ w\sup(A)]
               ∪ inf(B)
 Let sup(A v B) := [sup(A) ∪ sup(B)]
               ∩ sup(A v B)
 Let inf(A v B) := inf(A) ∪ inf(B)
               ∪ inf(A v B)

For each subgraph connecting A, B and A -> B:
 Let sup(A) := [w \ (inf(A->B) ∩ w\sup(B))]
               ∩ sup(A)
 Let inf(A) := (w \ sup(A->B)) ∪ inf(A)
 Let sup(B) := sup(A->B) ∩ sup(B)
 Let inf(B) := [inf(A->B) ∩ inf(A)] ∪ inf(B)
 Let sup(A->B) := [w\inf(A) ∪ sup(B)]
               ∩ sup(A->B)
 Let inf(A-B) := w\sup(A) ∪ inf(B)
               ∪ inf(A->B)

If for any formula, A, inf(A) ⊈ sup(A) then
exit with INCONSISTENCY FOUND
```

Since w is finite and there are only a finite number of formulae to be considered and the sup and inf values change monotonically, then the algorithm will eventually terminate.

On the problem above the algorithm makes the following calculations:

sup(A) = {a}

inf(A) = {a}

sup(A->B) = {a, b}
inf(A->B) = {a, b}

sup(B) = w
inf(B) = {}

sup(A) = [w \ ({a, b} ∩ w\w)] ∩ {a} = {a}
inf(A) = (w \ {a, b}) ∪ {a} = {c, a}

inf(A) $\subsetneq$ sup(A)
Therefore, exit with INCONSISTENCY FOUND

If the incidences of the axioms are restricted to be w or {} then the inconsistency detection algorithm degenerates into an incomplete algorithm for detecting contradictions in standard Propositional Logic and, hence, degenerates into an incomplete tautology checker. For instance, it will find an inconsistency if i(A&~A) is assigned w, but it will fail to detect the inconsistency if i(A->B), i(B->A), i(A v B) and i(~(A&B)) are all assigned w. Therefore, the algorithm is also incomplete for Propositional Incidence Logic.

A complete inconsistency detection algorithm is possible, but prohibitively expensive in this application. It would have to degenerate into a complete tautology checker. Note that tautology checkers for standard Propositional Logic are NP complete because they must consider every assignment of the two truth values to each proposition. A complete inconsistency detector for Propositional Incidence Logic would have to consider every assignment of the $2^{n(w)}$ possible incidences to each proposition.

In Predicate Incidence Calculus no complete terminating algorithm exists. Such an algorithm would have to degenerate into a complete terminating algorithm for detecting contradictions in standard Predicate Calculus, and this task is known to be semi-decidable. However, we can extend the incomplete inconsistency detection algorithm with the following steps.

**Definition 2:** Extensions to the Inconsistency Detection Algorithm:

Add arcs, to the dependency graph, connecting each formula, A, to each of its instances, A'.

Add the following constraint propagation rules:

```
Let sup(A) = sup(A') ∩ sup(A)
Let inf(A') = inf(A) ∪ inf(A')
```

The storage of sup(A) and inf(A) rather than i(A) is very similar to the storage of the interval [Spt(A), Pls(A)] in Shafer-Dempster theory rather than the probability, p(A). In fact,

Spt(A) = n(inf(A))/n(w) and

Pls(A) = n(sup(A))/n(w)

So Incidence Calculus can easily be adapted to provide a mechanism for dealing with the problem of dependent formulae in Shafer-Dempster theory, instead of in probability theory.

## 9 SOME MORE IMPLEMENTATION SUGGESTIONS

To initialize an incidence based system, axioms must be given and incidences must be assigned to them. In expert systems like MYCIN the initial assignment of numerical 'uncertainty factors' is made by the user. We will assume that users are prepared to assign probabilities (numbers) to axioms, but not incidences (sets). In this case, our task is to convert probabilities into incidences. Since incidences incorporate more information about the formulae than probabilities, namely the degree of independence of the formulae, we must either make assumptions about this extra information or provide a mechanism for the user to input it. We will take the former course, and assume that each axiom is as independent of the others as is allowed by the equations of Incidence Calculus. However, the mechanisms could be easily adapted to take into account user-supplied, correlation information.

In the case of Propositional Incidence Calculus the inconsistency detection algorithm can be re-activated on each incoming axiom to maintain dynamic lower and upper bounds for the assignment of incidences to axioms, i.e.

inf(A) ⊆ i(A) ⊆ sup(A)

Thus the assignment mechanism given below can be used.

**Definition 3:** Incidence Assignment Mechanism:

(a) If necessary, add A to the dependency graph and run the inconsistency detection algorithm to termination.

(b) Assign all elements in inf(A) to

i(A).

(c) Assign elements randomly to i(A) from sup(A) \ inf(A) to bring i(A) to a size such that p(A)=n(i(A))/n(w). *

(d) If step (c) was possible then reactivate the inconsistency detection algorithm. Fail if it finds an inconsistency.

(e) If the steps (c) or (d) cannot be executed then, if possible, back-up and reassign elements to previous incidences to correct the situation.

(f) If step (e) is not possible then complain to the use that his/her assignment of probabilities is inconsistent and request a reassignment.

For instance, suppose A is assigned the probability .75, B the probability .25, and A v B the probability 1, in that order. We will let w = {0,1,...,99}. The incidence assignment mechanism will start by building a dependency graph consisting only of A, with inf(A)={} and sup(A)=w. Step (b) is a non-op. Suppose step (c) assigns {0,1,...,74} to i(A). Step (d) will then assign {0,1,...74} to sup(A) and inf(A). Since steps (c) and (d) were successful then the remaining steps are not run.

Suppose that the mechanism assigns {0,4,8,...,96} to i(B), in a similar way. The trouble comes when it makes an assignment to i(A v B). Step (a) adds the first arcs to the dependency graph, and causes the first, non-trivial constraint propagation. sup(A v B) and inf(A v B) both get the value {0,1,2,...74,76,...,96}. The attempt, at step (c), to assign w to i(A) then fails. Step (e) forces reconsideration of the random assignment of {0,4,8,...96} to i(B). The only assignment that will work at this stage is {75,76,...,99}.

Note that the problem only arises if the assignment to i(A v B) is done last. If it is done before the assignment to i(A) or i(B) then the inconsistency detection algorithm will propagate constraints on these assignments which will force step (c) to assign unproblematic values to them. For instance, suppose i(A) is assigned a value as above and then [(A v B] is dealt with. Step (c) will assign w to both sup(A v B) and inf(A v B). Step (d) will assign {75,76,...,99} to inf(B). When step (b) works on B, it will assign all of {75,76,...,99} to i(B) and step (c) will find no further assignment is necessary to bring i(B) to the required size. This assignment will pass all remaining tests.

The perfection of the incidence assignment mechanism is one of the main outstanding problems for Incidence Calculus. In particular, step (e) above is only crudely specified, and it is not known whether the incompleteness of the inconsistency detection algorithm will cause problems in practice.

Back-up might be avoided in most cases by requesting from the user, not just probabilities for the formulae, but correlations between them. This information could then be used in step (c) to improve the random assignment of incidents to formulae. For instance, if the user had specified a correlation of -1 between A and B in the example above then they would have been assigned disjoint incidences in the first place.

Note that, in existing expert systems, the uncertainty values attached to all but some ground, propositional axioms (i.e. some 'facts') are set by the knowledge engineer. A similar assumption about Incidence Calculus based systems may enable them to guarantee consistency and avoid the need to remake the random assignment of incidents. Suppose we assume that the consistent incidences are assigned by the knowledge engineer before the system is run. This enables us to set up the dependency graph in advance with consistent assignments to sup and inf for all formulae in the non-propositional or non-ground axioms, i.e. to all formulae in the 'rules'. The user need only assign probabilities to the 'facts'. The inconsistency detection algorithm will only be run to propagate the incidencies of these 'facts' to the other axioms. I conjecture that this use of the algorithm is complete for Propositional Incidence Logic. The assignment of incidences to the 'rules' will force correlations on the 'facts' and limit the range of incidents that can be randomly assigned to them. I conjecture that any inconsistency that arises cannot be due to any faulty, random assignment in step (c) of the incidence assignment mechanism, but must be due to the inconsistent assignment of probabilities by the user. Thus the reassignment of incidents ( step (e)) will not be required.

The Incidence Assignment Mechanism could

*The accuracy of this random assignment technique can be improved by increasing the size of w.

be easily adapted to Shafer-Dempster theory by modifying steps (b) and (c) so that they changed the size of inf(A) and sup(A) so that they gave correct values for Spt(A) and Pls(A), rather than p(A).

## 10 CONCLUSION

In this note we have described a mechanism, Incidence Calculus, for incorporating probabilistic reasoning in an inference system. This mechanism is based on the assignment of sets to formulae, rather than the normal technique of assigning numbers to formulae. Our mechanism captures the properties of probabilistic reasoning (section 2), which a purely numeric mechanism could not do. It can be reasonably efficiently implemented using bit strings.

We intend to implement the mechanisms described in this paper and test them out in practice. Problems may arise because:

- the random assignment of incidents in step (c) of the incidence assignment mechanism may need to be remade;
- the incomplete inconsistency detection algorithm may fail to detect inconsistencies; and
- the provision, by the inference mechanism, of only a lower bound on the probabilities of formulae, may not prove strong enough in practice.

Of course, the last two problems are also present in *current* expert systems, and these systems do not have the benefit of assigning genuine probabilities to formulae. The first two problems may be avoidable by assuming that the incidences of all but ground propositions are set, correctly, by the knowledge engineer. The provision of only lower and upper bounds on the probabilities of formulae is made a virtue in Shafer-Dempster theory. Incidence Calculus can be readily adapted to this theory.

## ACKNOWLEDGEMENTS

I am grateful for comments and advice, on an earlier draft of this note, from Richard O'Keefe, David Spiegelhalter, Bernard Silver, Steve Owen and Alberto Petterossi. This research was supported by SERC grant, number GR/C/20826.

## REFERENCES

[Duda et at 78] Duda, R.O., Hart, P.E., Nilsson, N.J. and Sutherland, G.L.
Semantic network representations in rule-based inference systems.
In Waterman, D and Hayes-Roth, F. (editor), *Pattern- directed inference systems*, pages 203-221. Academic Press, 1978.

[Freund 72] Freund, J.E.
*Mathematical statistics.*
Prentice Hall, 1972.

[Lowrance & Garvey 82] Lowrance, J.D. and Garvey, T.D.
Evidential reasoning: A developing concept.
In *Proceedings of the International Conference on Cybernetics and Society*, pages 6-9. IEEE, 1982.

[Nilsson 84] Nilsson, N.
*Probabilistic Logic.*
Technical Note 321, SRI International, 1984.

[Shortliffe 76] Shortliffe, E.H.
*Computer-based medical consultations: MYCIN.*
North Holland, 1976.

[Zadeh 81] Zadeh, L.
PRUF - a meaning representational language for natural languages.
In Mamdani, E. and Gaines, B. (editors), *Fuzzy reasoning and its* applications, . Academic Press, 1981.