The Knowledge Engineering Review

Default non-monotonic logic

PETER MOTT

Department of Philosophy, University of Lancaster, Bailrigg, Lancaster

Abstract

This paper is a review of certain non-monotonic logics, which I call default non-monotonic logics. These are logics which exploit failure to prove. How each logic uses this basic idea is explained, and examples given. The emphasis is on leading ideas explained through examples: technical detail is avoided. Four non-monotonic logics are discussed: Reiter's default logic, McCarthy's circumscription, McDermott's modal non-monotonic logic, and Clarks's completed database. The first two are treated in some detail. The recent Hanks-McDermott criticism of non-monotonic logic is discussed, and some conclusions drawn about the prospects for non-monotonic logic. Recommendations for further reading are given.

1 Introduction

This paper aims to provide a review of certain non-monotonic logics, which I have called default non-monotonic logics. The paper is divided into four parts. The first part (sections 2 and 3) is a general introduction. I explain what default non-monotonic logics are for, what features make them non-monotonic, and how they are distinguished from other, not default but still non-monotonic, logics. The second part (section 4) gives an overview of four such logics: Reiter's default logic (1978, 1980), McCarthy's circumscription (1980, 1984, McDermott & Doyle's modal non-monotonic logic (1980, 1982), and Clark's negation as failure (1978). The third part (sections 5 and 6) gives an account in greater depth of two of the non-monotonic logics: Reiter's Default Logic and McCarthy's Circumscription. A basic dogma of default non-monotonic logic is that some not too radical adjustment of standard logic will provide a computationally useful representation of common sense reasoning. This assumption has recently been challenged in an important paper by Hanks and McDermott (1987) and the last part (section 7) is an analysis of their argument. In addition to its intrinsic interest this paper serves to illustrate the ideas previously introduced. I have included a final section of suggested reading.

I have tried to give the reader the leading ideas of these logics without becoming bogged down in technicalities. To this end I have confined the use of logical notation in the main text very severely (see Table 1 below). Anyone familiar with the PROLOG language should find they have enough knowledge of logic to understand what follows. There is of course a loss of detail in this approach which makes the logics look rather more like each other than they really are. But it does serve to

Table 1 Logical notations used

p,q,P,Q	any formulas
$X \vdash p$	the set of formulas X yields a proof of p
$q \leftarrow p$	q if p (material conditional)
$\neg p$	not p (negation)
pvq	p or$q$(disjunction)
$p\&q$	$p\&q$ (conjunction)
Mp	Possibly p (modal operator)

make a technical subject much more accessible than it otherwise would be, and anyway much of the technical detail may prove to be ephemeral although the underlying ideas, I am sure, will not.

2 The aim of default non-monotonic logic

The eventual aim of default non-monotonic logics is to solve the frame problem of artificial intelligence (Raphael, 1971; Minsky, 1975), a problem that arises as a consequence of using standard logic to represent real world knowledge. It may be, in fact it probably is, a deeper and more general problem as well, but that is beside the point here.

Suppose we represent Smith's knowledge of the world at a moment of time t1 by a set of facts in a PROLOG database including:

in(t1,door_closed)
in(t1,window_open)
in(t1,closes_window)
in(t1,light_on)

Time passes and it is soon t2. Smith needs new beliefs, and the question is how the update is to be managed. We can assume certain "causal laws" are available that relate facts at one moment to facts at the next moment. Such a law would be that if you close the window at a moment U then it will be closed at the next moment T:

in(T,window_closed) ← follows(T,U) & in(U,closes_window).

With the aid of this law we can get:

in(t2,window_closed)

But such causal laws say what changes, they are silent about what stays the same, while most of Smith's t2-knowledge is going to be of that sort. It will just be copied over from t1-knowledge. Amongst such knowledge we should expect both of the following since nothing has happened to change them:

in(t2,door_closed)
in(t2,light_on)

There are no "causal laws" that will let us draw these conclusions. Instead we have frame laws. These are a sort of "un-causal" law designed to fill the gap and record non-changes. We could have two such laws for the above situation, one saying that closing the window does not open the door, the other saying that closing the window does not make the light go out:

in(T,door_closed) ← follows(T,U) & in(U,closes_window)
in(T,light_on) ← follows(T,U) & in(U,closes_window)

Using these two laws we can copy the t1-knowledge over into t2-knowledge.

There are two difficulties with this approach via frame laws that matter here. The first is that if we add to t1-knowledge:

in(t1,opens_door)
in(T,door_open) ← follows(T,U) & in(U,opens_door).

we can derive:

in(t2,door_open)

But we can still, in logic, derive in(t2,door_closed) using the frame law. Somehow we want the addition of in(t1,opens_door) to the database to block the conclusion in(t2,door_closed) that we can draw in its absence. Frame laws will not deliver this and in fact we need a non-monotonic logic

as (as is explained in the next section) they are defined as logics where adding new premisses can block the derivation of old conclusions.

The second difficulty is that an approach through frame laws is wholly impractical since there would have to be too many of them. It compares with having to specify not only that 4 was the successor of 3 but that 0,1,2,5,6, . . . were not.

Non-monotonic logic offers a general form of solution to both aspects of this problem. In the manner of PROLOG we have instead of numerous frame laws, the single schematic law:

$$\text{in(T,F)} \leftarrow \text{in(U,F)} \;\&\; \text{follows(T,U)} \;\&\; \text{nothing in U ended F.}$$

A fact F will be assumed to persist into the next moment of time unless something definite disturbed it. This is the Galilean Law of Inertia generalised into a philosophical principle, a principle which Bertrand Russell once called "cosmic laziness", a principle that nothing changes unless it has to.

As it stands this is only the barest form of a solution. The non-monotonic logics variously attempt to give it substance.

It would be misleading to imply that every worker in non-monotonic logic has the frame problem in his mind all the time. Some no doubt think in even more general terms of the representation of common sense reasoning. Others have much more limited goals—for instance Clark who wanted to make sense of the behaviour of PROLOG's "not". But "common sense" is too broad to be much use in marking out the domain of non-monotonic logic, and justifying PROLOG's behaviour is too narrow. So I stand by the claim: the aim of non-monotonic logic is to solve the frame problem.

3 What makes a default non-monotonic logic

There is a simple condition that makes a logic (any logic) non-monotonic. Suppose that from a set of premisses X we can prove (in some given logic) a conclusion p. Suppose we now add further premisses Y. If it is guaranteed that p still follows from X and Y the logic is monotonic, otherwise it is non-monotonic. Non-monotonic logics are not new. In fact since the nineteenth century at least (Mill, 1843) logics have been divided into two sorts: deductive and inductive. Deductive logics are monotonic, inductive logics are non-monotonic.

In deductive logics a conclusion p is correctly drawn from a set of premisses X just when it is impossible that all the premisses in X be true and yet conclusion p false. Such a logic must be monotonic. For if it is impossible that all of X be true and p false, then it is also impossible for all of X and Y to be true and yet p false. Thus p is correctly concluded from the larger set of premisses $X+Y$ as well, so the logic is monotonic. A point that is often overlooked is that for a monotonic logic if X implies p then $X+\neg p$ implies p as well. No additional information at all can override a conclusion correctly drawn.

Inductive logics deal in probability, not certainty. In an inductive logic a conclusion p is correctly drawn from premisses X provided that the truth of X confirms, or makes probable the truth of p. And of course what is made probable by a certain amount of information may be made less so by more. Thus X may confirm p, but $X+Y$ may not. This means that inductive logics are non-monotonic.

There have been numerous inductive logics proposed (Kyburg, 1970) and all have the feature of being non-monotonic. In fact the literature is vast—the Kyburg volume has a thousand entry bibliography. Much of the effort has been spent around the philosophical problem of justifying inductive inference (Swinburne, 1974) which does not concern us here; but, as the reader hardly needs reminding, the use of logics of confirmation is extensive in knowledge engineering (Shortliffe, 1976), and the general issues of reasoning under uncertainty wider still (Cohen, 1987).

The question is how we can construe "non-monotonic logic" sufficiently narrowly to make it one thing rather than just a hotchpotch of many. For if the term covers everything from McCarthy's circumscription to the confirmation calculus in MYCIN, then it will be applied so widely as to be largely deprived of useful meaning. I propose then to distinguish default non-monotonic logic from logics of confirmation by the absence of any measure of degree of confirmation. If you see some-

thing like prob(h,e) = .7 then you are not looking at a default non-monotonic logic. Further, all the logics I will consider here seek to draw conclusions from the absence of information to the contrary. This is why I choose the name "default non-monotonic logic", because reasoning by default is just reasoning in the absence of information.

For computational purposes this focus is a natural one given that, as Reiter (1978) points out, the amount of negative information we hold far exceeds the amount of positive. For failure to prove a positive fact can be exploited in various ways to assert the complementary negative fact, with the obvious computational advantage that use of such a failure removes the need for the explicit storage of negative facts. Logics of this sort turn out to be non-monotonic because the addition of further information will reduce the number of things that cannot be proved and hence block old inferences.

4 Overview and introduction to default non-monotonic logics

In this section I will introduce four default non-monotonic logics: Reiter's closed world assumption (*CWA*), Clark's completed database (*CDB*), McDermott & Doyle's non-monotonic logic and McCarthy's circumscription.

4.a Reiter's closed world assumption (CWA)

A teaching timetable lists the classes taught, when they are taught and who teaches them. If there is no entry for Paul teaching maths in term 3 we conclude that he does not teach it then. This is not a logical consequence of anything the timetable explicitly asserts but rather of our implicit common sense assumption that what is not in the timetable is not taught. (Consider this as an instance of the law of cosmic laziness; people don't teach things until the timetable makes them.)

Given a PROLOG database *DB* comprising the assertions:

```
teaches(peter,maths,term3).
teaches(paul, english, term3).
teaches(mary,english,term1).
```

the query:

```
?—teaches(peter,maths,term1)
```

will cause PROLOG to respond with a "No". So PROLOG uses this simple database as we use a timetable (since "No" is the answer that we should give). PROLOG assumes that the timetable provides complete information, it implicitly adds negations to the database, in fact the negation of every assertion that is not actually in the database. This is known as the closed world assumption (*CWA*) (Reiter, 1978a).

To make the *CWA* for *DB* we add not only reasonable denials like:

```
¬teaches(peter,maths,term1),
¬teaches(paul,english,term1),
¬teaches(mary,maths,term1)
```

but also "silly" ones such as:

```
¬teaches(maths,maths,english).
¬teaches(term1,mary,english).
```

and so forth. In fact with the seven terms (logical, not academic) in the database there are 343 possible assertions about teaching. Three are asserted in the database, the negations of the other 340 are added by the *CWA*. If we conceive the database to be augmented with *CWA* then PROLOG's answer is a logical consequence of the *DB* + *CWA*.

Or almost. Suppose that we want to know if maths is taught in the Christmas term. It is only if someone teaches it so we query:

?—teaches(Person,maths,term1)

PROLOG will answer "No", but this is not a consequence of *DB*+*CWA*. The *CWA* includes the negations above, so it follows that neither Peter nor Paul nor Mary teach maths in the Christmas term. But the conclusion that PROLOG delivers is that there is no Person at all that teaches(Person,maths,term1) which is stronger than *CWA* since there may be such a person who is not named in the database. To exclude this requires the further Domain Closure Assumption (DCA) (Reiter, 1978b) that everything is named by some term of the database. When it is added then for this simple database, we have a logical theory that validates PROLOG's answer.

The basic idea of *CWA* and *DCA* is clearly that failure to deduce an assertion allows one to deduce its negation. The logic validates interpreters that behave in this way. Reiter's default logic, which we examine below, extends and develops this basic idea. This is obviously going to be non-monotonic, for add some assertion, say teaches(mary,maths,term1), and the *CWA* is correspondingly reduced.

4.b Clark's completed database (CDB)

Reiter's *CWA* implicitly adds the negation of literals that cannot be proved. Clark's completed database (*CDB*) (1978) (often called "Negation as failure") can be seen as a generalization of that idea. Everything you can do with *CWA* you can do with *CDB*, and the example of the previous section applies here as well.

Suppose we add to the teaching database the two further clauses:

teacher(P,science) ← teaches(P,maths,T).
teacher(P,science) ← teaches(P,physics,T).

and query:

?—teacher(paul,science).

PROLOG will answer "No", and once again this is not a logical consequence of the database. To see why, consider what a PROLOG interpreter would do here. Firstly it binds the variable P to paul and then sees if it can find some assertion teaches(paul,maths,T) in the database. It cannot, and the first clause fails. The process is repeated with the second clause and teaches(paul,physics,T). This also fails and, there being no more clauses to try, PROLOG answers "No".

The first logical blunder is the same as the previous case: because teaches(paul,maths,term1) is not in the database there can be no conclusion, in logic, that it is not true. But this may be met as before by using Reiter's *CWA*. So assume ¬teaches(paul,maths,term1) and the rest. It still doesn't follow. A clause $p \leftarrow q$ reads "q if p" so allows you to conclude p given q. Given $\neg q$ you fail to derive p, but this does not mean that you can derive $\neg p$. And likewise if there is another clause say $p \leftarrow r$, *then knowing* $\neg r$ still just blocks the derivation of p, rather than allowing the derivation of $\neg p$. And so on for $p \leftarrow s$, $p \leftarrow t$ and as many clauses as there are for p. The presence of the negations $\neg q, \neg r, \neg s, \neg t$ (courtesy of *CWA*) simply prevents the derivation of p. This failure however long continued never justifies the conclusion that $\neg p$.

The object of *CDB* is to add a sentence to the database that frees this blocked proof. Though rather technical to describe in full detail the idea is simple enough, just add the sentence: "if p then either q or r or s or t". For thus the proofs $\neg q, \neg r, \neg s, \neg t$ together deliver $\neg$(q or r or s or t) and hence $\neg p$. Using our example database we should get *CDB* by adding first all the negations of *CWA* and then the sentence:

teaches(P,physics,T) v teaches(P,maths,T) ← teacher(P,science)

This then justifies PROLOG's answer (it is not a horn clause of course)

4.c McDermott's modal non-monotonic logic

We next consider McDermott and Doyle's Modal non-monotonic logic. This is not representable in the syntax of PROLOG, though the enrichment required is, at least superficially, small. We introduce a modal operator M for "possible". This operator takes a sentence p and forms a sentence Mp which is read "It is possible that p". The operator M is in fact standard in modal logic and many different meanings are given to it. In fact for McDermott & Doyle Mp means "It is consistent to assume that p".

To see how this system works take again the teaching database. Add to it the following clauses, one for every assertion p that can be formulated in the language of the database:

$$\neg p \leftarrow M \neg p$$

thus, e.g. we have:

$$\neg \text{teaches(peter,maths,term1)} \leftarrow \text{M} \neg \text{teaches(peter,maths,term1)}$$
$$\neg \text{teaches(paul,english,term3)} \leftarrow \text{M} \neg \text{teaches(paul,english,term3)}$$

Such clauses will remain wholly inert unless there is some mechanism for concluding "possibles". The one that McDermott & Doyle supply is deceptively simple: the rule Poss. For any sentence at all (including negations):

(Poss) Conclude Mp if p is consistent with what you have concluded already.

Applied to our database this gives M$\neg p$ for all the negated assertions that can be consistently added to the database. Then the extra clause above delivers $\neg p$. And we have *CWA* again.

There are complications with this general approach which we shall look at in more detail when we consider Reiter's full default logic below (the two systems are quite similar). The immediate impression that McDermott's logic is more complicated is born out on a closer analysis.

4.d McCarthy's circumscription

With McCarthy's circumscription we must turn explicitly to the issue of common sense inference.

One of the standard examples of non-monotonic common sense inference seems to be:

Most birds can fly.
Tweety is a bird.
Tweety can fly

This argument could be made logically valid by replacing "most" with "all", but then the premiss is false. As it stands the premisses are true but the argument isn't valid. But since we use such inferences all the time, the question is how to extend logic to include them.

McCarthy's circumscription proposes the following answer. Suppose we represent the situation by the following clause:

$$\text{canfly(X)} \leftarrow \text{bird(X)} \,\&\, \neg \text{abnormal (X)}$$

This is true as far as it goes, but that is not far. We have no way of proving that tweety is not one of the abnormal birds. Circumscription introduces extra logical machinery aiming to enable us to do just that. The basic idea is that the predicate abnormal(X) should be "circumscribed", that is assumed to apply only to those things to which it must apply. But if we are not constrained to assume that tweety is abnormal we will assume that he is not.

One way to look at this is as asserting $\neg$abnormal(X) whenever we have not explicitly asserted abnormal(X) which is obviously similar to *CWA*. If we added clauses:

abnormal(X) ← penguin(X)
abnormal(X) ← ostrich(X)

and then used Clark's *CDB* we should be able to conclude that tweety could fly unless we had the explicit information that tweety was a penguin or an ostrich. And in fact circumscription, in simple cases, can be implemented using PROLOG. There is thus clearly a close relation between the logics we have been considering, which partly explains why in the next sections we concentrate on just two of them (Lifschitz, 1985a; Shepherdson, 1984).

5 Reiter's default logic

5.a Exposition of the logic

Reiter's default logic appeared in a long paper (1980) which in turn drew on his work on the *CWA*. I shall describe the logic and mention some difficulties in it, though only for the case of what Reiter calls "normal defaults". He also defines general defaults which are needed in some more complicated applications (Reiter and Criscuolo, 1983) but for all our purposes normal defaults will serve and in my view, general defaults introduce much intricacy with rather little to show for it.

A default is something that may be assumed from the absence of any information to the contrary provided its prerequisite is met. We can put:

X is a bird / X can fly

to indicate that the default conclusion "X can fly" may be drawn from the definite information "X is a bird" in the absence of definite information that "X cannot fly". Neither prerequisites nor defaults have to be simple sentences. We can, for example, have:

X is a bird v X is a bat v X is an aeroplane

as a prerequisite for:

X can fly ← (X is an aeroplane v ¬X is dead)

Though as seems so often the case the "natural" examples are the simple ones.

The question is then how we handle these defaults in Reiter's system. Once again the basic idea is failure to prove. That is, the absence of definite information to the contrary is modelled as failure to prove the contrary. So a default d may be concluded if we can prove its prerequisite and cannot prove its negation $\neg d$. The rule Poss to be added to the ordinary rules of logic is then:

From p / d, $\vdash p$ and not $\vdash \neg d$ infer d (Poss)

The elaboration of this idea is a lot more complicated than one might think. Before we face the difficulties let us see roughly how it should work. Suppose then that we have a bird database with defaults:

/ ¬ostrich(X)	;	its not an ostrich
bird(X)&¬ostrich(X) / flies(X)	;	birds fly, except ostriches
bird(tweety)	;	tweety is a bird

The first default has no prerequisite so it is always triggered. It amounts to assuming that something is not an ostrich before you know anything about it at all.

Thus we may infer:

¬ostrich(tweety)

which in turn gives us the prerequisite of the second default and hence we get flies(tweety). This means we end by extending the bird database to:

Table 2 *CWA* using defaults

The teaching database of section three was:

teaches(peter,maths,term3).
teaches(paul,english,term3).
teaches(mary,english,term1).

Add to this database the defaults $/\neg p$ for every assertion p that can be formulated in the language of the database. By the rule Poss we can infer $\neg p$ for all assertions except those where we can prove $\neg\neg p$. That is for all p except the three above. Hence we get *CWA* as a special case of the default logic.

bird(tweety)
¬ostrich(tweety)
flies(tweety)

But now suppose that we change the initial database to:

bird(tweety)
ostrich(tweety)

We can no longer add the default ¬ostrich(tweety) because we can already prove its negation ¬¬ostrich(tweety). So we can no longer meet the prerequisite of the second default and so cannot conclude flies(tweety). There is therefore no way of using the defaults to extend the new bird database. Thus Poss makes logic non-monotonic because adding new premisses blocks old conclusions.

There is an apparent conceptual difficulty in the use of the rule Poss: it appears circular. For in order to decide what we can infer with Poss we have to decide what prerequisites we can prove; but to do this we must know, among other things, what we can infer with Poss. To be more definite yet, look back at the definition of Poss just given and ask just what is the sense of $\vdash p$ there used? There are two possibilities. It means "the database yields p by the ordinary rules of logic" or it means "the database yields p by the ordinary rules of logic + the rule Poss". The second appears to define Poss by assuming it in the meaning of $\vdash$.

So what happens if we try with the obviously non-circular sense? What extra we can infer using Poss will then depend only on what we can and cannot infer standardly without it. This is straightforward but it will not work.

Recall the bird database: it should allow the conclusion that tweety can fly since he is a bird and there is no reason to think him an ostrich. But we can't prove this with the first meaning of $\vdash$. For to get flies(tweety) we have to prove standardly bird(tweety) & ¬ostrich(tweety). But to get ¬ostrich-(tweety) we need to use the rule Poss and that is not standard. So taking $\vdash$ to mean standard proof is too weak for a useful default logic. It seems that we need the second "circular" form of Poss, and therefore the question to which we now turn is how to understand the circularity in a way that is harmless.

Suppose we add to the bird database so that it is as below:

```
D1 / ¬ostrich(X)                    ; its not an ostrich
D2 bird(X)&¬ostrich(X) / flies(X)   ; birds except ostriches fly
D3 pet(X) / ¬flies(X)               ; most pets don't fly
 1 bird(tweety)                     ; tweety is a bird
 2 pet(tweety)                      ; tweety is a pet
```

We can derive flies(tweety) like this:

```
3 ¬ostrich(tweety)                  ; Poss, D1
4 bird(tweety)&¬ostrich(tweety)     ; 1 and 3
```

Table 3 The rule Poss

At a stage S in extending a database *DB* you can infer *d* from a default *p/d* using Poss provided that

(i) $\vdash p$ from S
(ii) not $\vdash \neg d$ from S

S + *d* is the next stage. The starting stage is *DB*.

A stage S at which nothing more can be inferred by Poss is an extension of *DB*. Extensions are fixed points of the rule Poss

5 flies(tweety) ; Poss, D2, prerequisite 4

But note that the line:

6 ¬flies(tweety) ; Poss, D3, prerequisite 2

is not now legitimate because we have already derived 5 and so can prove from the database as so far extended that ¬¬tweety(flies). Thus what the rule Poss allows depends on the stage we have reached in extending the original database by defaults.

There need be no unique extension. Suppose we began to extend the database by:

3 ¬flies(tweety) ; Poss, D3, prerequisite 2

This is quite legitimate. And we can continue as before:

4 ¬ostrich(tweety) ; Poss, D1.
5 bird(tweety)&¬ostrich(tweety) ; 1 and 4

But now the blocked line is:

6 flies(tweety) ; Poss, D2, prerequisite 5

for from the database as presently extended we can prove ¬flies(tweety). Again, what we can prove at any stage in extending a database depends upon what we have already proved, hence on previous selections of defaults. There is no circularity involved in Poss but instead a cumulative process dependent at any stage on what has gone before.

5.b Criticism of default logic

To finish the account of Reiter's default logic I shall mention two criticisms of it.

The first is that it has no obvious and natural semantics to guide the construction of reasonable defaults. Take, for example, the defaults:

its_raining/its_windy, its_windy/its_raining

What exactly is asserted by these? Are they reasonable? The only way to find out is by drawing out their consequences, that is procedurally, which is only feasible in simple cases. (This point should not be overstated. The CWA has a clear semantics, namely that what is not asserted is denied. Further Konolige (1988) has shown how to translate default theories into Moore's auto-epistemic logic (Moore, 1985) and this translation may be held to bring a semantics with it.)

Secondly, having got your defaults how do you decide which extension to use, for as the previous example showed, there are (often) several alternatives. I do not know of any principled answer to this question. The difficulty is that the choice of default seems to involve the application of the sort of understanding that we typically call common sense. And, as the theorem-proving community found out (Bundy 1983, p. 82ff), attempts to supplant common sense by formal devices however ingenious seem to succumb rather quickly to combinatorial explosion.

This last criticism is not peculiar to default logic but applies to all the logics we are considering.

Table 4 Circumscription and formal models

A model M = (D,I) of a database is a pair consisting of a set of objects D called the domain of the model and an interpretation function I defined for the language of the database.

Let *DB* be a database, $p(X_1 .. X_n)$ a predicate that occurs in D, and M a model of *DB*

Then M = (D,I) is a p-minimal model of *DB* if there is no model M' = (D',I') such that:

(i) D = D'

(ii) I = I' for each name and function symbol

(iii) I'(p) a proper subset of I(p).

A formula *Q* is a consequence of database *DB* by circumscription of *p* provided that *Q* is true in every p-minimal model of D.

The above is a definition of variable circumscription when all predicates are allowed to vary. Restrictions can be put on the predicates allowed to vary (see for example Mott 1987).

Non-monotonic logics do not come with instructions on how to use them sensibly. McCarthy's circumscription which we consider next does, however, have a semantics.

6 McCarthy's circumscription

6.a Exposition of circumscription

Circumscription (McCarthy 1980, 1984) is the most studied version of non-monotonic logic, partly because of the clarity of its basic semantic idea and partly, I suspect, because it is logically interesting. McCarthy suggests that common sense is characterized by its minimizing or "circumscribing" tendency. It tends to confine the extension (that is the range of application) of certain predicates as far as is possible to be consistent with what is known. Circumscription is McCarthy's attempt to find a logical representation of this process.

The idea behind circumscription was described informally in section 4.d. There is some difficulty in giving a more accurate account while avoiding technical details, largely because a full account involves the use of minimal models and no knowledge of model theory is assumed here. I shall therefore proceed using an informal conception of "model", a more precise formulation is given in Table 4.

Suppose we have the following bird database:

flies(X) ← bird(X)&¬abnormal(X)
abnormal(X) ← emu(X)
bird(tweety)

By a model of a database I mean an intuitive state of affairs involving various objects which makes all the formulas in the database true. A formula is a logical consequence of a database provided that the formula has to be true in all models of the database. For example flies(tweety) ← ¬abnormal(tweety) is a logical consequence of the bird database above since bird(tweety) has to be true in any model of the database. Pretty clearly flies(tweety) is not a logical consequence of the bird database. There are models of it where tweety does not fly because he is abnormal.

Now consider what is true given this database if we circumscribe abnormal(X). To do this we must admit only minimal models, that is we must make the database true in such a way that there are as few abnormal(X) things as possible. Whatever is true in all such minimal models (as opposed to all models) is said to follow from the database with abnormal(X) circumscribed.

In the bird database it seems clear that there need be no abnormal things at all—so minimal models have an empty extension of abnormal(X). Suppose we have such a model and see what must be true in it:

1 ¬abnormal(tweety) ; nothing is abnormal
2 abnormal(tweety) ← emu(tweety) ; a model of the database

Table 5 Syntactic definition of circumscription

Let A(P,Q) be a formula and P,Q the predicate constants occurring in A(P,Q). The circumscription of P in A with Q allowed to vary is the 2nd-order formula:

$$A \,\&\, \forall p \forall q(A(p,q) \,\&\, \forall x(px \rightarrow Px) \rightarrow \forall x(Px \rightarrow px)). \qquad \text{(Circ)}$$

where A(p,q) comes from A by replacing predicate constants by predicate variables. (This definition is generalized to several "variable" predicates $Q_1 \ldots Q_n$ and to n-ary predicates $Px_1 \ldots x_n$ in the obvious way).

Circ says that if p, q are chosen so as to satisfy the formula A and p is at least as small as P then p and P are the same size.

Suppose that A is Pa & Pb. We show that:

1 $\forall x(Fx \rightarrow x = a \vee x = b)$ follows from the circumscription of F in A (there are no "variables"). Choose the predicate $x = a \vee x = b$ for px in (Circ) to obtain:

2 $a = a \vee a = b \,\&\, b = a \vee b = b \,\&\, \forall x(x = a \vee x = b \rightarrow Px) \rightarrow \forall x(Px \rightarrow x = a \vee x = b)$

But the first two conjuncts are logical truths and the third is a consequence of Pa&Pb. Thus we can detach 1.

By circumscribing P we have shown that there are at most two things that are P. The formula A is naturally taken as the conjunction of formulas in a database.

3 ¬emu(tweety) ; logic from 1,2
4 flies(tweety) ← bird(tweety)&¬abnormal(tweety) ; a model of database
5 bird(tweety) ; a model of database
6 flies(tweety) ; logic from 1,4,5

So in minimal models of the bird database tweety can fly and is not an emu. These are then consequences of the database by circumscription of abnormal(X).

This makes a non-monotonic logic. For suppose that we add to the bird database the item emu(tweety). A minimal model will now have tweety as the only abnormal thing, for in any such model:

1 emu(tweety) ; a model of database
2 abnormal(tweety) ← emu(tweety) ; a model of the database
3 abnormal(tweety) ; logic from 1,2

But we cannot conclude that tweety flies. In some minimal models he does and in others he does not. In fact minimizing abnormal(X) no longer gives any new information about tweety at all. Thus the addition of further premisses blocks old conclusions and the logic is non-monotonic.

The above gives some insight into the semantic definition of circumscription. In McCarthy's presentations, the approach is however syntactic: you circumscribe by adding new axioms to the database (somewhat in the manner which we have already encountered with Reiter's *CWA* and Clark's *CDB*). This formulation is technical and need not delay us here (see Table 5). The central ideas of circumscription are captured by the semantic account.

There have been certain technical difficulties with circumscription (in some forms it is inconsistent) and there is a more general doubt about how best to include the semantic insights into a working system. This latter is a mirror-image of the difficulty with Reiter's default logic. Default logic seems to lack a natural semantics, circumscription seems to lack a natural proof procedure. However, rather than dwell on these issues here I shall move at once to consideration of the Hanks-McDermott problem, for that poses some rather serious questions about the viability of default non-monotonic logics in general. It will also serve as an extended example of the application of circumscription to the frame problem.

7 The Hanks-McDermott problem

In this section I will describe in detail an example of the (attempted) application of circumscription to the frame problem. This was given by Hanks & McDermott (1987) in support of a general criti-

cism of circumscription in particular and non-monotonic logic in general (it also holds for Reiter's default logic and McDermott's own logic). They claim that non-monotonic logics are too weak to formalize common sense reasoning, a weakness which they attribute not to any particular defect of the logics concerned but to their general reliance on simple syntactic or semantic devices. Such they conjecture, will always fail to capture anything like the subtlety of common sense inference. This criticism is obviously important in itself for it amounts to denying that there is any logic to common sense, but an analysis of their example will also given an opportunity for a more substantial illustration of the ideas of non-monotonic logic than we have so far had.

A conclusion p follows from the circumscription of a database only if p is true in all minimal models of the database. What is true in some minimal models but not in others is thus not a consequence. Hanks & McDermott give an example where the common sense conclusion is only true in one minimal model, not all. So circumscription cannot derive it.

6.a The Hanks-McDermott murder story—a simple version

The murderer loads his gun, he pauses briefly, he shoots his victim in the heart. This is the story we have to represent. The conclusion we want to derive, using circumscription, is that the victim is eventually dead.

Hanks and McDermott use a version of the situation calculus to represent the murder story. The representation that follows is somewhat simpler than theirs and is confined to horn clauses so that it could be run by a PROLOG interpreter.

The language of the murder story has just the three binary predicate symbols:

in, endsIn, follows.

and the names:

0,1,2,3
alive, dead, loaded, unloaded, loads, shoots, pauses

The former represent situations or moments of time, the latter facts or events which may be true in a situation (or at a moment). We put:

in(T,F)

to mean that the fact F is true in (at) the situation T. A fact F that is true in T1 may cease to be true in the following moment T2, in which case we put:

endsIn(T,F)

Situations have only one important feature: they succeed each other. So follows(T2,T1) meaning that situation T2 comes after situation T1.

After some preprocessing, the facts of the case can be given as follows:

```
in(0,unloaded)   ;  the gun is unloaded
in(0,alive)      ;  the victim is alive
in(0,loads)      ;  the murderer loads the gun
in(1,pauses)     ;  he pauses briefly
in(2,shoots)     ;  he shoots the victim
```

To draw conclusions we supply some "causal laws";

in(T,loaded) ← follows(T,U) & in(U,loads)
in(T,dead) ← follows(T,U) & in(U,loaded) & in(U,shoots)
endsIn(T,alive) ← in(T,loaded) & in(T,shoots)

The common sense consequence we want to get out of the system is that the victim dies: in(3,dead). From our earlier remarks about the frame problem we are going to need more than the

laws of things changing, and we do not attempt to put in all the "laws" about things that do not change. Instead we will adopt the law of the inertia of states of affairs that says that things go on as they were unless something disturbs them. In fact what we naturally call "states of affairs" satisfy this law, while what we naturally call "events" do not. Loading a gun ends an unloaded condition, shooting someone with a loaded gun ends their alive condition. But unless and until such events occur the gun remains unloaded and the person alive. On the other hand we think of events differently—no law of inertia applies to them. In our model we take events to be "instantaneous" and to end in the situation in which they occurred. Thus we add:

endsIn(T,E) ← in(T,E) & event(E).
event(loads).
event(shoots).
event(pauses).

The law of inertia of states of affairs is our frame law, which is now expressed as follows:

in(T,Fact) ← follows(T,U) & in(U,Fact) & ¬endsIn(U,Fact)

The frame law says that Fact will hold at a moment T if Fact held at the previous moment and nothing happened then to end Fact.

6.b The frame problem: failure of non-monotonic logic

There is quite a lot of information represented above, yet it is not enough to prove that the victim dies. This is because we need the frame law to project facts forward in time. But in order to use it to prove in(T,F) we need to be able to prove ¬endsIn(U,F). But in general this is not a logical consequence of our representation. We cannot prove the negations we need from the mere absence of information, and in general such an absence of information is all that we have to go on.

One can see how circumscription in particular and non-monotonic logic in general, ought to help. Circumscription minimizes predicates as far as possible, so if we circumscribe endsIn we should have ¬endsIn(U,F) provable in the circumscribed theory except where we have explicit information to the contrary.

But, and this is what Hanks-McDermott showed, it doesn't work. Circumscription will not deliver the goods. Instead there are two possibilities compatible with the axioms we have set up, and in only one of them is the victim dead. In the other the murderer's pausing unloads the gun! We will see how this turns out next, but for the moment consider how acutely embarrassing this is for McCarthy's proposal. The circumscriptive engine founders on the possibility that maybe the gun unloaded itself while the murderer paused and this is the very sort of idiot outcome that circumscription was originally (McCarthy, 1980) advertised to exclude.

Neither is Reiter's default logic or McDermott's non-monotonic logic in any better position. Both permit the two models: that the victim died or the gun unloaded itself. (In Reiter's terms there are two extensions of the murder story.)

In the next two sections I shall set out in greater detail just how circumscription fails to deliver the consequence we expect.

6.c Two models of the murder story

Table 6 below shows two models of the murder story in diagram form.

It directly diagrams the in(T,F) predicate. To see whether in(T,F) holds one simply reads the T row along to the F column. If the entry is "y" then it does, if "n" then it does not (the two models are the same except where the second model has an entry in parentheses). So, for example, we have in(0,loads) but not in(2,pauses). The first model has in(3,dead), the second model not.

The endsIn predicate is determined by the table as well: endsIn(T,F) is true just in case F is true in T but becomes false in T + 1. In other words endsIn marks the transitions of a given state of affairs

Table 6 Two models of the murder story

t	unloaded	loads	loaded	pauses	shoots	alive	dead
0	y	y	n	n	n	y	n
1	n	n	y	y	n	y	n
2	n (y)	n	y (n)	n	y	y	n
3	n (y)	n	y (n)	n	n	n (y)	y (n)

Model 2 in brackets when it differs from Model 1.

from true to false. We have endsIn(1,pauses) in both models. In model 1 but not model 2 we have endsIn(2,alive). In model 2 but not model 1 we have endsIn(1,loaded). Note that we interpret endsIn precisely so that the frame law is true.

This leaves only the interpretation of follows(T,U) which is obvious: follows(T,U) if and only if $T = U + 1$. So follows means happens at the next moment, and moments are interpreted as numbers.

Both models are models of the murder story. To show this in detail would be very tedious, so I will only give a couple of examples beginning with the frame law.

Proposition 1. The frame law is true in both models

Pf. Suppose follows(T,U) & in(U,F) & ¬endsIn(U,F). Since ¬endsIn(U,F) there is no transition from true to false at U for F. But since in(U,F) this means that in(U + 1,F). Finally from follows(T,U) we have $T = U + 1$). Thus in(T,F).

Inspection of the table shows that explicitly dated events are all true in both models. Likewise the causal laws which can be shown by the same sort of argument that was just given for the frame law. Spare the details. For the endsIn statements we need only check that there is an appropriate change of truth-value. Again I give one example:

Proposition 2. endsIn(T,loads) ← In(T,loads) is true in both models.

Pf. Inspection of the table shows that $T = 0$ is the only value that satisfies in(T,loads). But since ¬In(2,loads) we conclude endsIn(1,loads). So the law is true.

6.d Adding circumscription makes no difference

Both models make true the murder story, and hence that there can be no logical consequence that in(3,dead). This is not surprising of course, what we now show is that circumscription makes no difference.

To make effective use of the frame law we circumscribe endsIn(T,F) according to the commonsense assumption that states of affairs like being loaded, continue the same unless something definite happens to end them.

It is quite simple to check whether this delivers the consequence that in(3,dead). It will do so provided that in every model minimal with respect to endsIn, in(3,dead) is true. Consequently circumscription will fail to deliver that result if Model 2 is a minimal model of the murder story. This in turn can be determined by seeing if it is possible to squeeze down the predicate endsIn in Model 2 while keeping a model of the murder story. And it is not possible as the following argument shows.

Inspection of Table 6 shows that there are five endsIn statements true in Model 2:

endsIn(0,loads)
endsIn(0,unloaded)
endsIn(1,pauses)
endsIn(2,shoots)
endsIn(1,loaded)

Of these, the first four are logical consequences of the database, so they must be true in any model of it minimal or not. That leaves the last. But if we set it false then we can prove endsIn(2,alive). So

Table 7 Proof that Model 2 is minimal

1	in(2,loaded)← follows(2,1) & in(1,loaded) & ¬endsIn(1,loaded)	;	Frame Law
2	follows(2,1)	;	Fact
3	in(1,loaded) ← in(0,loads)	;	Causal Law
4	in(0,loads)	;	Fact
5	in(1,loaded)	;	by 3,4
6	¬endsIn(1,loaded)	;	Assumption
7	in(2,loaded)	;	by 1,2,5,6
8	endsIn(2,alive) ← in(2,loaded) & in(2,shoots)	;	Causal Law
9	in(2,shoots)	;	Fact
10	endsIn(2,alive)	;	by 7,8,9

removing one endsIn simply constrains another, and thus Model 2 is minimal. Essentially one simply oscillates between the two minimal models indicated: all that follows by circumscription is what is true in both of them and that is not what we want. The reader who wants to check the argument in detail can consult Table 7.

8 Conclusion—what are the prospects for non-monotonic logic?

The murder story of Hanks & McDermott is important because it gives grounds for doubting the basic enterprise of non-monotonic logic. They write:

"... there is no reason to expect that a simple syntax for describing when 'leaps of faith' should be made, or a simple minimality criterion such as set inclusion, should handle all cases."

And the point is fairly made that these logics all assume that there is something fundamentally simple and tractable underlying common sense reasoning, and in all cases so far the assumed something has proved inadequate.

Further, Hanks & McDermott argue that circumscription is no good at actually deriving conclusions, only at showing that those otherwise derived are correct:

"So far the only way to use circumscribed theories has been to perform analyses such as the one we performed above: you know what conclusion you want to get out of the theory, you use the conclusion to build an instance of the circumscription axiom, and you see if the desired result pops out."

This criticism is extended to other non-monotonic logics: they are able to tell when a conclusion may be legitimately inferred by their canons, but give you no help in inferring it in the first place.

Does all this mean that non-monotonic logic is something to be abandoned, even that (as Hanks & McDermott originally suggested) logic is a poor representation language for AI? I think not. Logic is far too big and too protean to be criticized *en masse*, and anyway there is no alternative available. My conclusion is somewhat different. Non-monotonic logic has probably concerned itself too much with validating inferences and not enough with making them. That calls for a change of emphasis to a more computationally and less philosophically based approach—but hardly the abandonment of a paradigm.

That said, Hanks & McDermott's criticism must not be overstated. I set up the murder story so that it conformed to the syntax of PROLOG. If you actually run it you will find that PROLOG will choose the right answer: it works, the victim dies. Why? Obviously the PROLOG interpreter circumscribes it's predicates, but it apparently does more, it chooses a particular minimal model. I do not know how to characterize the additional processes involved—the point is that though minimizing predicates is not on its own sufficient to deliver the required common sense result, it seems that mixed with some other process it is. Why should this not be a general phenomenon? Common sense reasoning may be the cumulative effect of numerous simple mechanisms like circumscription, severally intelligible with plausible semantics, jointly sufficient to the task at hand. It is legitimate to study

them one by one—the only error would consist in expecting any single mechanism to be by itself a magic key for unlocking the mysteries of common sense reasoning.

My own doubts about default non-monotonic logic are rather different. The computational processes connected with current non-monotonic logics appear to be cumbersome, and there are technical results that show this is not accidental for logics based around the failure to prove (see Schlipf, 1987; Kunen, 1987; Fitting, 1985; Blair, 1982). Furthermore standard logic was not invented to simulate common sense reasoning but to render absolutely precise the language of mathematics. Such precision became essential after the discovery of the logical paradoxes at the turn of the century: precision by giving explicit axioms was no longer enough, the language itself had to be made exact and explicit. But the complexities that resulted are not those that are found in common sense reasoning. For example consider the following argument:

> Every person who has received any benefit for more than six months is entitled to receive at least long-term supplementary rate. Mary Smith has received widow's benefit for a year therefore Mary Smith is entitled to at least long-term supplementary rate.

This is patently a good argument. I do not say that it cannot be formalized in standard logic, only that any formalization would make the proof of the conclusion from the premisses long, indirect and difficult. What is more, most people would actually draw the conclusion if presented with the premisses and asked about Mary Smith, and would do so without difficulty. For them the argument is easy, for predicate logic complex. This is a poor prospect for a logic of common sense reasoning, non-monotonic or not.

It seems to me very likely that some formalism not based upon standard logic is going to be necessary—one which finds such arguments like the above example easy. Perhaps the situation theory being developed by Jon Barwise and others (Barwise and Perry, 1983; Barwise and Etchmendy, 1987) will eventually open new possibilities for applications.

9 Suggested Reading

These notes are intended to assist the reader who wants to know more about non-monotonic logic. I have included material on circumscription and negation in PROLOG, since in my judgment these are the active areas, and also some references to more general issues. Reiter's default logic is an elegant construction indeed and grew out of "real" problems but it is probably now a closed chapter.

9.a Logic texts

Gibbins (1988) expounds standard logic using PROLOG. A very uncluttered introduction is Boolos & Jeffrey (1982). Mendelson (1987) is comprehensive, utterly reliable and great value for money.

9.b Circumscription

The papers by McCarthy (1980, 1984) are the original sources. They are far from clear. Lifschitz (1985a) is very useful, relating circumscription to the *CWA*. Davis (1980) has some basic results. He showed that circumscription could lead to contradiction; Lifschitz (1986) and Mott (1987) are about keeping circumscription consistent. Perlis & Minker (1986) discuss how far circumscription produces all the conclusions it should. Hanks & McDermott (1987) we have already discussed.

The logical prerequisites for circumscription are quite extensive. For 2nd-order logic see Boolos & Jeffrey (1980) or Van Dalen (1985). Harder to come by but very good is Robbin (1969). Circumscription is often presented using the lambda calculus. I know of no succinct introduction. Barendregt (1984) is a treatise on the subject. For model theory, Bridge (1977) is clear and short if one can tolerate a lot of symbols. Mendelson (1987) is free of excessive notation and full, the standard treatise is Chang and Keisler (1976) which in its earlier parts is quite accessible. Tarski (1969, 1944) are semi-popular expositions of the basic idea of model theory by the man who created it. The technical question of how far circumscription is computationally tractable is addressed in Schlipf (1987a).

9.c Negation as failure

We can be quite brief since Loyd (1984) is a full text on the theory of PROLOG. Negation as failure was introduced in Clark (1978). Recent technically difficult discussions are Fitting (1985), Kunen (1987) and Van Gelder (1988). The relation between Reiter's logic and negation as failure is treated in Shepherdson (1984, 1985). These two papers are fully argued and more accessible. There is a sustained conceptual criticism of negation as failure in Flannagan (1986). The logical prerequisites are supplied by Lloyd (1984).

9.d Related logic

Non-monotonic logic is a species of inductive logic, an introduction to which is Kyburg (1970). For the frame problem the logic of time is obviously important. Van Benthem (1983) and Rescher & Urquhart (1971) describe temporal logics. The standard introduction to modal logic (of which temporal logic is a species) remains Hughes & Cresswell (1968, and later editions). Hilpinen (1981) is a book of papers on deontic logic, another variety of logic that has found some application in artificial intelligence. Finally Turner (1984) is a general introduction to the uses of non-standard logics in artificial intelligence. Some texts on non-monotonic logic appear to be imminent, but none are yet available as far as I know. Apart from the main journals the recently introduced bulletin of applied non-classical logics contains abstracts of papers and books recently published in this area. It is available from Dr L. Cerro, Universite/Paul Sabatier, Languages et Systemes Informatique, 118 route de Narbonne, 31062 Toulouse Cedex—France.
(e_mail:mcvax!inria!geocub!farinas)

Acknowledgement

I thank John Fox and a referee for criticism of a previous draft.

Appendix

PROLOG program of Hanks McDermott murder story

The version of the Hanks McDermott murder story in the paper conforms to the syntax of PROLOG. It is reproduced here with some auxiliary predicates which create Table 6, Model 1 of the murder story. I hope it may be of some use to teachers. To use the program call story. A call story(N) will extend the story for times up to N. As things are, nothing more happens after 3pm (post mortem). It will be easy to add other laws and events to the database.

```
/*FACTS
in(0,unloaded).
in(0,alive).
in(0,loads).
in(1,pauses).
in(2,shoots).

/*CAUSAL LAWS
in(X,loaded):- follows(X,X1), in (X1,loads).
in(X,dead):- follows(X,X1), in(X1,loaded), in(X1,shoots).
endsIn(X,unloaded:- in(X,loads).
endsIn(X,alive):- in(X,loaded), in(X,shoots).

/*EVENTS
/*Events don't persist but begin and end at the same time */
endsIn(X,E):- event(E),in(X,E).
```

```
event(loads).
event(shoots).
event(pauses).

/*FRAME LAW                                                    */
in(Time,Fact):-
        follows(Time,PreviousTime),
        in(PreviousTime,Fact),
        not( endsIn(PreviousTime,Fact) ).

/*Time is the series of the integers                           */
follows(X,Y):- integer(X), X>0, Y is X - 1.

/*PROCEDURES TO BUILD MODEL 1 of the Murder Story              */
story:- story(3).
story(N):-
        write ('In(T,F): unloaded loads loaded pauses shoots alive dead'), nl,
write('--------:----------------------------'),
        nl,
        dotimes(0,N).
dotimes(Low,High):-
        Low > High,
        nl.
dotimes(T,N):-
        write('  '), write(T), write('  :'),
        isin(T,unloaded,Ansl), tab(5), write(Ansl),
        isin(T,loads,Ans2), tab(7), write(Ans2),
        isin(T,loaded,Ans3), tab(6), write(Ans3),
        isin(T,pauses,Ans4), tab(6), write(Ans4),
        isin(T,shoots,Ans5), tab(6), write(Ans5),
        isin(T,alive,Ans6), tab(5), write(Ans6),
        isin(T,dead,Ans7), tab(5), write(Ans7),
        nl,
        T1 is T + 1,
        dotimes(T1,N).
isin(N,X,Ans):-
        in(N,X), Ans = 'y'.
isin(N,X,Ans):-
        Ans = 'n'.
```

References

Barendregt, H 1981. *Lambda Calculus*, North Holland, Amsterdam.

Barwise, J and Etchmendy, J, 1987. *Truth and the Liar*, Oxford.

Barwise, J and Perry, J, 1983. *Situations and Attitudes*, MIT Press.

Bossu, G and Siegal, P, 1985. "Saturation, non-monotonic reasoning and the closed world assumption", *Artificial Intelligence* (25) 13–64.

Boolos, G and Jeffrey, R, 1982. *Computability & Logic*, Cambridge.

Blair, H A, 1982. "The recursion-theorectic complexity of the semantics of predicate logic as a programming language," *Information and Control* (54) 25–47.

Bridge, J, 1977. *Beginning model theory*, Oxford Logic Guides, Oxford.

Bundy, A, 1983. *The Computer Modelling of Mathematical Reasoning*, Academic Press, London.

Chang, C C and Keisler, H J, 1976. *Model Theory*, North-Holland.

Clark, K L, 1978. "Negation as failure," In: Gallaire & Minker op cit 293.

Cohen, P, 1987. "The control of reasoning under uncertainty: a discussion of some programs," *The Knowledge Engineering Review* (2) 6–25.

Davis, M, 1980. "The mathematics of non-monotonic reasoning," *Artificial Intelligence* (13) 73–80.
Etherington, D, Mercer, R and Reiter, R, 1985. "On the adequacy of predicate circumscription for closed world reasoning," *Computational Intelligence* (1) (1985) 11–15.
Fitting, M, 1985. "A Kripke-Kleene semantics for logic programs," *Journal of Logic Programming* (4) 295–312.
Fitting, M, 1986. "Notes on the mathematical aspects of Kripke's theory of truth," *Notre Dame Journal of Formal Logic* (27) 75–88.
Flannagan, T, 1986. "The consistency of negation as failure," *Journal of Logic Programming* (3) 93–115.
Gabbay, D, 1982. "Intuitionistic bases for non-monotonic logic," In: *Proc. Conference on Automated Deduction Springer Lecture Notes in Computer Science* (6) 260–273.
Gallaire, H and Minker, J, (Ed.), 1978. *Logic and Databases*, Plenum Press, New York.
Gibbins, P, 1988. Logic with Prolog (Oxford).
Hanks, S and McDermott, D, 1987. "Nonmonotonic logic and temporal projection," *Artificial Intelligence* (33) 379–412.
Hilpinen, R, (ed.), 1981. *New Studies in Deontic Logic*, North Holland, Amsterdam.
Hughes, G E and Cresswell, M, 1968. *An Introduction to Modal Logic*, Methuen, London.
Jaffar, J, Lassez, J-L and Lloyd, J W, 1983. "Completeness of the negation as failure rule, "*Proceedings of the 8th IJCAI*, Karlsruhe, 500–506.
Konolige, K, 1988. "On the relation between default and autoepistemic logic," *Artificial Intelligence* (35) 343–382.
Kunen, K, 1987. "Negation in logic programming," *Journal of Logic Programming* (4) 289–308.
Kyburg, H, 1970. *Probability and Inductive Logic*, Macmillan, London.
Lifschitz, V, 1985a. "Closed world data bases and circumscription," *Artificial Intelligence* (27) 229–335.
Lifschitz, V, 1985b. "Pointwise circumscription: preliminary report," *Proc. IJCAI-85*, Los Angeles, Ca., 406–10.
Lifschitz, V, 1986. "On the satisfiability of circumscription," *Artificial Intelligence* (28) 17–28.
Lloyd, J W, 1984. *Foundations Of Logic Programming*, Springer-Verlag, Berlin.
McCarthy, J, 1980. "Circumscription—a form of non-monotonic reasoning," *Artificial Intelligence* (13) 27–39.
McCarthy, J, 1986. "Applications of circumscription to formalising common-sense knowledge," *Artificial Intelligence* (28) 89–116.
McDermott, D V and Doyle, J, 1980. "Non-monotonic logic I," *Artificial Intelligence* (13) 41–72.
McDermot, D V, 1983. "Non-monotonic logic II," *JACM* (29) 33–57.
McDermott D V, 1982. "A temporal logic for reasoning about processes and plans," *Cognitive Sci.* (6) 101–155.
Mendelson, E, 1987. *Introduction to Mathematical Logic* (3rd Edition), Wadsworth, California.
Minsky, M, 1975. "A framework for representing knowledge," P Winston (Ed.), *The Psychology of Computer Vision* McGraw Hill, New York.
Mill, J S, 1843. *A System Of Logic.*
Moore, R C, 1985. "Semantic considerations on non-monotonic logic," *Artificial Intelligence* (25) 75–94.
Mott, P L, 1987. "A theorem on the consistency of circumscription," *Artificial Intelligence* (31) 87–98.
Nute, D, 1988. "LDR: a logic for defeasible reasoning," *ACMC Research Report 01–0013*, University of Georgia.
Perlis, D and Minker, J, 1986. "Completeness results for circumscription," *Artificial Intelligence* (28) 29–42.
Popper, K R, 1972. *Objective Knowledge*, OUP, London.
Raphael, B, 1971. "The frame problem in problem solving systems," In: N V Findler & B Meltzer (Eds.), *AI and Heuristic Programming*, Edinburgh U P, Edinburgh.
Reiter, R, 1980a. "A logic for default reasoning," *Artificial Intelligence* (13) 81–132.
Reiter, R, 1980b. "Equality and domain closure in first order data bases," *JACM* (27) 235–249.
Reiter, R, 1978." On closed world data bases," In: *Gallaire & Minker*, op cit 55–77.
Reiter, R, 1978b. "Deductive question answering on relational databases," In: *Gallaire and Minker*, op cit 149–177.
Reiter, R and Crisculo, G. 1983. "Some representational issues in default reasoning," *Int. J. of Computat. Maths* (9) 1–13.
Rescher, N and Urquhart, A, 1971. *Temporal Logic*, Springer Verlag, Berlin.
Robbin, J W, 1969. *Mathematical Logic*, Benjamin, New York.
Schlipf, J, 1987a. "Decidability and definability with circumscription," *Annals of Pure and Applied Logic* (35) 173–191.
Schlipf, J, 1987b. "When is closed world reasoning tractable?" Department of Computer Science, University of Cincinati, Ohio 45221–0008, USA.
Shepherdson, J, 1984. "Negation as failure: a comparison of Clark's completed database and Reiter's closed world assumption," *Journal of Logic Programming* (1) 51–79.
Shepherdson, J, 1985. "Negation as failure II," *Journal of Logic Programming* (2) 185–202.
Shortliffe, E, 1976. *Computer-Based Medical Consultations: MYCIN*, Elsevier, New York.

Smets, Ph, Mamdani, E H, Dubois, D and Prade, H, 1988. *Non-Standard Logics for Automated Reasoning*, Academic Press, London.
Swinburne, R (Ed), 1974. *The Justification of Induction*, Oxford.
Tarski, A, 1944. "The semantic conception of truth" *Philosophy & Phenomenological Research* **4** 341–376.
Tarski, A, 1969a. *Logic, Semantics, Metamathematics*, J H Woodger, (Ed.), Oxford.
Tarski, A, 1969b. "Truth and proof," *Scientific American* **194** (6).
Turner, R, 1984. *Logics for Artificial Intelligence*, Ellis Horwood, New York.
Van Benthem, J, 1983. *The Logic of Time*, Reidel, Dordrecht.
Van Dalen, D, 1985. *Logic and Structure*, Springer, Berlin.
Van Gelder, A, Ross, K and Schlipf, J S, 1988. "Unfounded sets and well-founded semantics for general logic programs," *7th ACM Principles of Database Systems.*
Wojcik, M, 1987. "Methods of automatic reasoning using non-monotonic logics—examples of implementation", Polish Academy of Sciences, ICS Report 610, Warsaw.