ON REASONING BY DEFAULT

Raymond Reiter

Department of Computer Science
University of British Columbia
Vancouver, B.C., Canada

ABSTRACT

This paper surveys a number of kinds of default reasoning in Artificial Intelligence, specifically, default assignments to variables, the closed world assumption, the frame default for causal worlds, exceptions as defaults, and negation in Artificial Intelligence programming languages. Some of these defaults provide clear representational and computational advantanges over their corresponding first order theories. Finally, the paper discusses various difficulties associated with default theories.

If I don't know I don't know
I think I know
If I don't know I know
I think I don't know

R.D. Laing, Knots

1. INTRODUCTION

Default reasoning is commonly used in natural language understanding systems and in Artificial Intelligence in general. We use the term "default reasoning" to denote the process of arriving at conclusions based upon patterns of inference of the form "In the absence of any information to the contrary, assume..." In this paper, we take this pattern to have the more formal meaning "If certain information cannot be deduced from the given knowledge base, then conclude..." Such reasoning represents a form of plausible inference and is typically required whenever conclusions must be drawn despite the absence of total knowledge about a world.

In order to fix some of these ideas, we begin by surveying a number of instances of default reasoning as they are commonly invoked in A.I. Specifically, we discuss default assignments to variables, the closed world assumption, the frame default for causal worlds, exceptions as defaults, and negation in A.I. programming languages. We shall see that these may all be formalized by introducing a single default operator $\not\vdash$ where $\not\vdash W$ is taken to mean "W is not deducible from the given knowledge base".

In addition, we shall discover that the closed world and frame defaults provide clear representational and computational advantages over their corresponding first order theories. The former eliminates the need for an explicit representation of negative knowledge about a world, while the latter eliminates the so-called frame axioms for dynamic worlds.

Finally, we discuss various problems which arise as a result of augmenting first order logic with a default operator.

2. SOME INSTANCES OF DEFAULT REASONING IN A.I.

The use of default reasoning in A.I. is far more widespread than is commonly realized. The purpose of this section is to point out a variety of seemingly different situations in which such reasoning arises, to accent common patterns which emerge when defaults are formalized, and to indicate certain representational and computational advantages of default reasoning.

2.1 Default Assignments to Variables

A number of knowledge representation schemes, e.g. FRL [Roberts and Goldstein 1977], KRL [Bobrow and Winograd 1977], explicitly provide for the assignment of default values to variables (slots, terminals). For example, in KRL the unit for a person in an airline travel system has the form:

```
[Person UNIT Basic
        .
        .
        .
<hometown{(a City) PaloAlto; DEFAULT}>
        .
        .
        .  ]
```

We can view this declaration as an instruction to the KRL interpreter to carry out the following:
If x is a person, then in the absence of any information to the contrary, assume hometown(x)=PaloAlto, or phrased in a way which makes explicit the fact that a default assignment is being made to a variable:
If x is a person and no value can be determined for the variable y such that hometown(x)=y, then assume y=PaloAlto.
Notice that in assigning a default value to a variable, it is not sufficient to fail to find an explicit match for the variable in the data base. For example, the non existence in the data base of a fact of the form hometown(JohnDoe)=y for some city y does not necessarily permit the default assignment y=PaloAlto. It might be the case that the following information is available:

$$(x/\text{EMPLOYER})(y/\text{PERSON})(z/\text{CITY})\text{EMPLOYS}(x,y) \wedge \text{location}(x)=z \supset \text{hometown}(y)=z$$ [1]

i.e. a person's hometown is the same as his or her employer. In this case the default assignment y=PaloAlto can be made only if we fail to deduce the existence of an employer x and city z such that

$$\text{EMPLOYS}(x,\text{JohnDoe}) \wedge \text{location}(x)=z$$

In general then, default assignments to variables are permitted only as a result of failure of some attempted deduction. We can formulate a general inference pattern for the default assignment of values to variables:
For all $x_1,\ldots,x_n$ in classes $\tau_1,\ldots,\tau_n$ respectively, if we fail to deduce $(Ey/\theta)P(x_1,\ldots,x_n,y)$ then infer the default statement

$$P(x_1,\ldots,x_n,\langle\text{default value for } y\rangle)$$

or more succinctly,

$$\frac{(x_1/\tau_1)\ldots(x_n/\tau_n) \quad \not\vdash (Ey/\theta)P(x_1,\ldots,x_n,y)}{P(x_1,\ldots,x_n,\langle\text{default value for } y\rangle)} \qquad (D1)$$

Here $\not\vdash$ is to be read "fail to deduce", θ and the τ's are types, and $P(x_1,\ldots,x_n,y)$ is any statement about the variables $x_1,\ldots,x_n,y$. There are some serious difficulties associated with just what exactly is meant by "$\not\vdash$" but we shall defer these issues for the moment and rely instead on the reader's intuition. The default rule for home towns can now be seen as an instance of the above pattern:

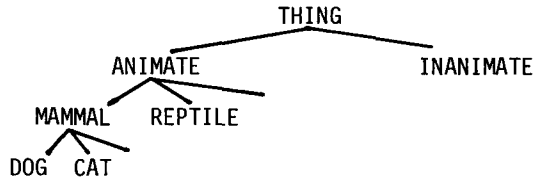
$$(x/\text{PERSON})\ \frac{\not\vdash (Ey/\text{CITY})\text{hometown}(x)=y}{\text{hometown}(x)=\text{PaloAlto}}$$

2.2 THE CLOSED WORLD ASSUMPTION

It seems not generally recognized that the reasoning components of many natural language understanding systems have default assumptions built into them. The representation of knowledge upon which the reasoner computes does not explicitly indicate certain default assumptions. Rather, these defaults are realized as part of the code of the reasoner, or, as we shall say, following [Hayes 1977], as part of the reasoner's process structure. The most common such default corresponds to what has elsewhere been referred to as the closed world assumption [Reiter 1978]. In this section we describe two commonly used closed world defaults.

2.2.1 Hierarchies

As an illustration of the class of closed world defaults, consider standard taxonomies (IS-A hierarchies) as they are usually represented in the A.I. literature, for example the following:



This has, as its first order logical representation, the following:

[1] Throughout this paper we shall use a typed logical representation language. Types, e.g. EMPLOYER, PERSON, CITY correspond to the usual categories of IS-A hierarchies. A typed universal quantifier like (x/EMPLOYER) is read "for all x which belong to the class EMPLOYER" or simply "for all employers x". A typed existential quantifier like (Ex/CITY) is read "there is a city x". The notation derives from that used by Woods in his "FOR function" [Woods 1968].

$$\left.\begin{array}{l}(x)DOG(x) \supset MAMMAL(x)\\ (x)CAT(x) \supset MAMMAL(x)\\ (x)MAMMAL(x) \supset ANIMATE(x)\\ \text{etc.}\end{array}\right\} \qquad (2.1)$$

Now if Fido is known to be a dog we can conclude that Fido is animate in either of two essentially isomorphic ways:

1. If the hierarchy is implemented as some sort of network, then we infer ANIMATE(fido) if the class ANIMATE lies "above" DOG i.e. there is some pointer chain leading from node DOG to node ANIMATE in the network.
2. If the hierarchy is implemented as a set of first order formulae, then we conclude ANIMATE(fido) if we can forward chain (modus ponens) with DOG(fido) to derive ANIMATE(fido). This forward chaining from DOG(fido) to ANIMATE(fido) corresponds exactly to following pointers from node DOG to node ANIMATE in the network.

Thus far, there is no essential difference between a network representation of a hierarchy with its pointer-chasing interpreter and a first order representation with its forward chaining theorem proving interpreter. A fundamental distinction arises with respect to negation. As an example, consider how one deduces that Fido is <u>not</u> a reptile. A network interpreter will determine that the node REPTILE does not lie "above" DOG and will thereby conclude that DOGs are not REPTILEs so that ¬REPTILE(fido) is deduced. On the other hand, a theorem prover will try to prove ¬REPTILE(fido). Given the above first order representation, no such proof exists. The reason is clear - nothing in the representation (2.1) states that the categories MAMMAL and REPTILE are disjoint. For the theorem prover to deal with negative information, the knowledge base (2.1) must be augmented by the following facts stating that the categories of the hierarchy are disjoint:

$$\left.\begin{array}{l}(x)ANIMATE(x) \supset \neg INANIMATE(x)\\ (x)MAMMAL(x) \supset \neg REPTILE(x)\\ (x)DOG(x) \supset \neg CAT(x)\end{array}\right\} \qquad (2.2)$$

It is now clear that a first order theorem proving interpreter can establish ¬REPTILE(fido) by a pure forward chaining proof procedure from DOG(fido) using (2.1) and (2.2). However, unlike the earlier proof of ANIMATE(fido), this proof of ¬REPTILE(fido) is <u>not isomorphic to that generated by the network interpreter</u>. (Recall that the network interpreter deduces ¬REPTILE(fido) by <u>failing</u> to find a pointer chain linking DOG and REPTILE). Moreover, while the network interpreter must contend only with a representation equivalent to that of (2.1), the theorem prover must additionally utilize the negative information (2.2). Somehow, then, the process structure of the network interpreter implicitly represents the negative knowledge (2.2), while computing only on declarative knowledge equivalent to (2.1).

We can best distinguish the two approaches by observing that two different logics are involved. To see this, consider modifying the theorem prover so as to simulate the network process structure. Since the network interpreter tries, and fails, to establish a pointer chain from DOG to REPTILE using a declarative knowledge base equivalent to (2.1), the theorem prover can likewise attempt to prove REPTILE(fido) using only (2.1). As for the network interpreter, this attempt will fail. If we now endow the theorem prover with the additional inference rule:

"If you fail to deduce REPTILE(fido) then conclude ¬REPTILE(fido)"

the deduction of ¬REPTILE(fido) will be isomorphic to that of the network interpreter. More generally, we require an inference schema, applicable to any of the monadic predicates MAMMAL, DOG, CAT, etc. of the hierarchy:

"If x is an individual and P(x) cannot be deduced, then infer ¬P(x)"

or in the notation of the previous section

$$(x)\ \frac{\nvdash P(x)}{\neg P(x)} \qquad (D2)$$

What we have argued then is that the process structure of a network interpreter is formally equivalent to that of a first order theorem prover augmented by the ability to use the inference schema (D2). In a sense, a network interpreter is the compiled form of such an augmented theorem prover.

There are several points worth noting:

1. The schema (D2) is not a first order rule of inference since the operator $\nvdash$ is not a first order notion. (It is a meta notion.) Thus a theorem

prover which evokes (D2) in order to establish negative conclusions by failure is not performing first order deductions.

2. The schema (D2) has a similar pattern to the default schema (D1).

3. In the presence of the default schema (D2), the negative knowledge (2.2), which would be necessary in the absence of (D2), is not required. As we shall see in the next section, this property is a general characteristic of the closed world default, and leads to a significant reduction in the complexity of both the representation and processing of knowledge.

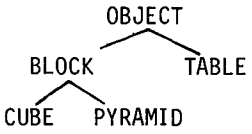
2.2.2 The Closed World Default

The schema (D2) is actually a special case of the following more general default schema:

$$(x_1/\tau_1)\ldots(x_n/\tau_n)\ \frac{\not\vdash P(x_1,\ldots,x_n)}{\neg P(x_1,\ldots,x_n)} \qquad \text{(D3)}$$

If (D3) is in force for all predicates P of some domain, then reasoning is being done under the closed world assumption [Reiter 1978]. In most A.I. representation schemes, hierarchies are treated as closed world domains. The use of the closed world assumption in A.I. and in ordinary human reasoning extends beyond such hierarchies, however. As a simple example, consider an airline schedule for a direct Air Canada flight from Vancouver to New York. If none is found, one assumes that no such flight exists. Formally, we can view the schedule as a data base, and the query as an attempt to establish DIRECTLY-CONNECTS(AC, Van,NY). This fails, whence one concludes ¬DIRECTLY-CONNECTS(AC,Van,NY) by an application of schema (D3). Such schedules are designed to be used under the closed world assumption. They contain only positive information; negative information is inferred by default. There is one very good reason for making the closed world assumption in this setting. The number of negative facts vastly exceeds the number of positive ones. For example, Air Canada does not directly connect Vancouver and Moscow, or Toronto and Bombay, or Moscow and Bombay, etc. etc. It is totally unfeasible to explicitly represent all such negative information in the data base, as would be required under a first order theorem prover. It is important to notice, however, that the closed world assumption presumes perfect knowledge about the domain being modeled. If it were not known, for example, whether Air Canada directly connects Vancouver and Chicago, we would no longer be justified in making the closed world assumption with respect to the flight schedule. For by the absence of this fact from the data base, we would conclude that Air Canada does not directly connect Vancouver and Chicago, violating our assumed state of ignorance about this fact.

The flight schedule illustrates a very common use of the closed world default rule for purely extensional data bases. In particular, it illustrates how this default factors out the need for any explicit representation of negative facts. This result holds for more general data bases. As an example, consider the ubiquitous blocks world, under the following decomposition hierarchy of objects in that world:



Let SUPPORTS(x,y) denote "x directly supports y" and FREE(x) denote "x is free" i.e. objects may be placed upon x. Then the following general facts hold:

$$(x/\text{OBJECT})(y/\text{TABLE})\neg\text{SUPPORTS}(x,y) \qquad (1)$$

$$(x/\text{OBJECT})\neg\text{SUPPORTS}(x,x) \qquad (2)$$

$$(x/\text{PYRAMID})(y/\text{BLOCK})\neg\text{SUPPORTS}(x,y) \qquad (3)$$

$$(x\ y/\text{BLOCK})\text{SUPPORTS}(x,y) \supset \neg\text{SUPPORTS}(y,x) \qquad (4)$$

$$(x/\text{PYRAMID})\neg\text{FREE}(x) \qquad (5)$$

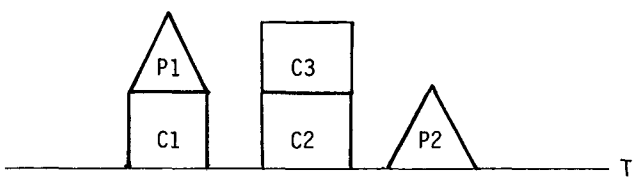
$$(x\ y/\text{BLOCK})(z/\text{TABLE})\text{SUPPORTS}(x,y) \supset \neg\text{SUPPORTS}(z,y) \qquad (6)$$

$$(x/\text{CUBE})\text{FREE}(x) \supset (y/\text{BLOCK})\neg\text{SUPPORTS}(x,y) \qquad (7)$$

$$(x/\text{CUBE})(y/\text{BLOCK})\neg\text{SUPPORTS}(x,y) \supset \text{FREE}(x) \qquad (8)$$

$$(x/\text{TABLE})\text{FREE}(x) \qquad (9)$$

Consider the following scene



This is representable by

$$\left.\begin{array}{ll} \text{SUPPORTS(T,C1)} & \text{SUPPORTS(T,C2)} \\ \text{SUPPORTS(C1,P1)} & \text{SUPPORTS(C2,C3)} \\ \text{SUPPORTS(T,P2)} & \end{array}\right\} \quad (10)$$

together with the following negative facts

$$\left.\begin{array}{ll} \neg\text{SUPPORTS(C1,C2)} & \neg\text{SUPPORTS(C2,C1)} \\ \neg\text{SUPPORTS(C3,C1)} & \neg\text{SUPPORTS(C1,P2)} \\ \neg\text{SUPPORTS(C3,P1)} & \neg\text{SUPPORTS(C3,P2)} \\ \neg\text{SUPPORTS(C1,C3)} & \neg\text{SUPPORTS(C2,P1)} \end{array}\right\} \quad (11)$$

Notice that virtually all of the knowledge about the blocks domain is negative, namely the negative specific facts (11), together with the negative facts(1)-(7)[1]. This is not an accidental feature. Most of what we know about any world is negative.

Now a first order theorem prover must have access to all of the facts (1)-(11). For example, in proving ¬SUPPORTS(C3,C2) it must use (4). Consider instead such a theorem prover endowed with the additional ability to interpret the closed world default schema (D3). Then, in attempting a proof of ¬SUPPORTS(C3,C2) it tries to show that SUPPORTS(C3,C2) is not provable. Since SUPPORTS(C3,C2) cannot be proved, it concludes ¬SUPPORTS(C3,C2), as required.

It should be clear intuitively that in the presence of the closed world default schema (D3), none of the negative facts (1)-(7), (11) need be represented explicitly nor used in reasoning. This can be proved, under fairly general conditions [Reiter 1978]. One function, then, of the closed world default is to "factor out" of the representation all negative knowledge about the domain. It is of some interest to compare the blocks world representation (1)-(11) with those commonly used in blocks world problem-solvers (e.g.[Winograd 1972, Warren 1974]). These systems do not represent explicitly the negative knowledge (1)-(7), (11) but instead use the closed world default for reasoning about negation. (See Section 3 below for a discussion of negation in A.I. programming languages.)

Although the closed world default factors out negative knowledge for answering questions about a domain, this knowledge must nevertheless be available. To see why, consider an attempted update of the example blocks world scene with the new "fact" SUPPORTS(C3,C2). To detect the resulting inconsistency requires the negative fact (4). In general then, negative knowledge is necessary for maintaining the integrity of a data base. A consequence of the closed world assumption is a decomposition of knowledge into positive and negative facts. Only positive knowledge is required for querying the data base. Both positive and negative knowledge are required for maintaining the integrity of the data base.

2.3 DEFAULTS AND THE FRAME PROBLEM

The frame problem [Raphael 1971] arises in the representation of dynamic worlds. Roughly speaking, the problem stems from the need to represent those aspects of the world which remain invariant under certain state changes. For example, moving a particular object or switching on a light will not change the colours of any objects in the world. Painting an object will not affect the locations of the objects. In a first order representation of such worlds, it is necessary to represent explicitly all of the invariants under all state changes. These are referred to as the frame axioms for the world being modeled. For example, to represent the fact that painting an object does not alter the locations of objects would require, in the situational calculus of [McCarthy and Hayes 1969] a frame axiom something like

$$(x\ z/\text{OBJECT})(y/\text{POSITION})(s/\text{STATE})(C/\text{COLOUR})$$
$$\text{LOCATION}(x,y,s) \supset \text{LOCATION}(x,y,\text{paint}(z,C,s))$$

The problem is that in general we will require a vast number of such axioms e.g. object locations also remain invariant when lights are switched on, when it thunders, when someone speaks etc. so there is a major difficulty in even articulating a deductively adequate set of frame axioms for a given world.

A solution to the frame problem is a representation of the world coupled with appropriate rules of inference such that the frame axioms are neither represented explicitly nor used explicitly in reasoning about the world. We will focus on a

[1] The notion of a negative fact has a precise definition. A fact is negative iff all of the literals in its clausal form are negative.

proposed solution by [Sandewall 1972][1]. A related approach is described in [Hayes 1973]. Sandewall proposes a new operator, UNLESS, which takes formula W as argument. The intended interpretation of UNLESS(W) is "W can not be proved" i.e. it is identical to the operator $\not\vdash$ of this paper. Sandewall proposes a single "frame inference rule" which, in the notation of this paper, can be paraphrased as follows:

For all predicates P which take a state variable as an argument

$$(x_1/\tau_1)\ldots(x_n/\tau_n)(s/\text{STATE})(f/\text{ACTION-FUNCTION})\quad \frac{\not\vdash \neg P(x_1,\ldots,x_n,f(x_1,\ldots,x_n,s))}{P(x_1,\ldots,x_n,f(x_1,\ldots,x_n,s))} \qquad \text{(D4)}$$

Intuitively, (D4) formalizes the so-called "STRIPS assumption" [Waldinger 1975]: Every action (state change) is assumed to leave every relation unaffected unless it is possible to deduce otherwise. This schema can be used in the following way, say in order to establish that cube33 is at location λ after box7 has been painted blue:

To establish LOCATION(cube33,λ,paint(box7,blue,s))
fail to prove $\neg$LOCATION(cube33,λ,paint(box7,blue,s))

There are several observations that can be made:

1. The frame inference schema (D4) has a pattern similar to the default schemata (D2) and (D3) of earlier sections of this paper. It too is a default schema.
2. The frame schema (D4) is in some sense a dual of the closed world schema (D3). The former permits the deduction of a positive fact from failure to establish its negation. The latter provides for the deduction of a negative fact from failure to derive its positive counterpart. This duality is preserved with respect to the knowledge "factored out" of the representation. Whereas the frame default eliminates the need for certain kinds of positive knowledge (the frame axioms), the closed world default factors out the explicit representation of negative knowledge.

2.4 DEFAULTS AND EXCEPTIONS

A good deal of what we know about the world is "almost always" true, with a few exceptions. For example, all birds fly except for penguins, ostriches, fledglings, etc. Given a particular bird, we will conclude that it flies unless we happen to know that is satisfies one of these exceptions. Nevertheless, we want it true of birds "in general" that they fly. How can we reconcile these apparently conflicting points of view? The natural first order representation is inconsistent:

(x/BIRD)FLY(x) "In general, birds fly"
(x)PENGUIN(x) $\supset$ BIRD(x) "Penguins are birds
(x/PENGUIN)$\neg$FLY(x) which don't fly."

An alternative first order representation explicitly lists the exceptions to flying

$$(x/\text{BIRD})\neg\text{PENGUIN}(x) \wedge \neg\text{OSTRICH}(x) \wedge \ldots \supset \text{FLY}(x)$$

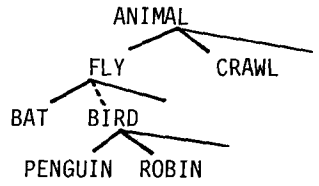
But with this representation, we cannot conclude of a "general" bird, that it can fly. To see why, consider an attempt to prove FLY(tweety) where all we know of tweety is that she is a bird. Then we must establish the subgoal

$$\neg\text{PENGUIN}(\text{tweety}) \wedge \neg\text{OSTRICH}(\text{tweety}) \wedge \ldots$$

which is impossible given that we have no further information about tweety. We are blocked from concluding that tweety can fly even though, intuitively we want to deduce just that. In effect, we need a default rule of the form

$$(x/\text{BIRD})\quad \frac{\not\vdash (\text{PENGUIN}(x) \vee \text{OSTRICH}(x) \vee \ldots)}{\text{FLY}(x)}$$

With this rule of inference we can deduce FLY(tweety), as required. Notice, however, that whenever there are exceptions to a "general" fact in some domain of knowledge we are no longer free to arbitrarily structure that knowledge. For example, the following hierarchy would be unacceptable, where the dotted link indicates the existence of an exception



Clearly there is no way in this hierarchy of establishing that penguins are animals. For hierarchies the constraint imposed by exceptions is easily

[1] [Kramosil 1975] claims to have proved that Sandewall's approach is either meaningless or equivalent to a first order approach. See Section 4 for a discussion of this issue.

articulated: If P and Q are nodes with P below Q, and if $(x)P(x) \supset Q(x)$ is true without exception, then there must be a sequence of solid links connecting P and Q. For more general kinds of knowledge the situation is more problematic. One must be careful to ensure that chains of implications do not unwittingly inherit unintended exceptions.

3. DEFAULTS AND "NEGATION" IN A.I. PROGRAMMING LANGUAGES

It has been observed by several authors [Hayes 1973, Sandewall 1972, Reiter 1978] that the basic default operator $\not\vdash$ has, as its "procedural equivalent" the negation operator in a number of A.I. programming languages e.g. THNOT in MICROPLANNER [Hewitt 1972, Sussman et al. 1970], NOT in PROLOG [Roussel 1975]. For example, in MICROPLANNER, the command (THGOAL <pattern>) can be viewed as an attempt to prove <pattern> given a data base of facts and theorems. (THNOT(THGOAL <pattern>)) then succeeds iff (THGOAL <pattern>) fails i.e. iff <pattern> is not provable, and this of course is precisely the interpretation of the default operator $\not\vdash$.

Given that "negation" in A.I. procedural languages corresponds to the default operator and not to logical negation, it would seem that some of the criticism often directed at theorem proving from within the A.I. community is misdirected. For the so-called procedural approach, often proposed as an alternative to theorem proving as a representation and reasoning component in A.I. systems, is a realization of a default logic, whereas theorem provers are usually realizations of a first order logic, and as we have seen, these are different logics.

In a sense, the so-called procedural vs. declarative issue in A.I, might better be phrased as the default vs. first order logic issue. Many of the advantages of the procedural approach can be interpreted as representational and computational advantages of the default operator. There is a fair amount of empirical evidence in support of this point of view, primarily based upon the successful use of PROLOG [Roussel 1975] - a pure theorem prover augmented with a "THNOT" operator - for such diverse A.I. tasks as problem solving [Warren 1974], symbolic mathematics [Kanoui 1976], and natural language question-answering [Colmerauer 1973].

On the theoretical level, we are just beginning to understand the advantages of a first order logic augmented with the default operator:

1. Default logic provides a representation language which more faithfully reflects a good deal of common sense knowledge than do traditional logics. Similarly, for many situations, default reasoning corresponds to what is usually viewed as common sense reasoning.
2. For many settings, the appropriate default theories lead to a significant reduction in both representational and computational complexity with respect to the corresponding first order theory. Thus, under the closed world default, negative knowledge about a domain need not explicitly be represented nor reasoned with in querying a data base. Similarly under the frame default, the usual frame axioms are not required.

There are, of course, other advantages of the procedural approach - specifically, explicit control over reasoning - which are not accounted for by the above logical analysis. We have distinguished the purely logical structure of such representational languages from their process structure, and have argued that at least some of their success derives from the nature of the logic which they realize.

4. SOME PROBLEMS WITH DEFAULT THEORIES

Given that default reasoning has such widespread applications in A.I. it is natural to define a default theory as a first order theory augmented by one or more inference schemata like (D1), (D2) etc. and to investigate the properties of such theories. Unfortunately, some such theories display peculiar and intuitively unacceptable behaviours.

One difficulty is the ease with which inconsistent theories can be defined, for example $\frac{\not\vdash A}{B}$ coupled with a knowledge base with the single fact $\neg B$. Another, pointed out by [Sandewall 1972] is that the theorems of certain default theories will depend upon the order in which they are derived. As an example, consider the theory

$$\frac{\not\vdash A}{B} \qquad \frac{\not\vdash B}{A}$$

Since A is not provable, we can infer B. Since B

is now proved, we cannot infer A, so this theory has the single theorem B. If instead, we had started by observing that B is not provable, then the theory would have the single theorem A. Default theories exhibiting such behaviour are clearly unacceptable. At the very least, we must demand of a default theory that it satisfy a kind of Church-Rosser property: No matter what the order in which the theorems of the theory are derived, the resulting set of theorems will be unique.

Another difficulty arises in modeling dynamically changing worlds e.g. in causal worlds or in text understanding where the model of the text being built up changes as more of the text is assimilated. Under these circumstances, inferences which have been made as a result of a default assumption may subsequently be falsified by new information which now violates that default assumption. As a simple example, consider a travel consultant which has made the default assumption that the traveller's starting point is Palo Alto and has, on the basis of this, planned all of the details of a trip. If the consultant subsequently learns that the starting point is Los Angeles, it must undo at least part of the planned trip, specifically the first (and possibly last) leg of the plan. But how is the consultant to know to focus just on these changes? Somehow, whenever a new fact is deduced and stored in the data base, all of the facts which rely upon a default assumption and which supported this deduction must be associated with this new fact. These supporting facts must themselves have their default supports associated with them, and so on. Now, should the data base be updated with new information which renders an instance of some default rule inapplicable, delete all facts which had been previously deduced whose support sets relied upon this instance of the default rule. There are obviously some technical and implementation details that require articulation, but the basic idea should be clear. A related proposal for dealing with beliefs and real world observations is described in [Hayes 1973].

One way of viewing the role of a default theory is as a way of implicitly further completing an underlying incomplete first order theory. Recall that a first order theory is said to be complete iff for all closed formulae W, wither W or $\neg$W is provable. Most interesting mathematical theories turn out to be incomplete - a celebrated result due to Godel. Most of what we know about the world, when formalized, will yield an incomplete theory precisely because we cannot know everything - there are gaps in our knowledge. The effect of a default rule is to implicitly fill in some of those gaps by a form of plausible reasoning. In particular, the effect of the closed world default is to fully complete an underlying incomplete first order theory. However, it is well known that there are insurmountable problems associated with completing an incomplete theory like arithmetic. Although it is a trivial matter conceptually to augment the axioms of arithmetic with a default rule $\frac{\not\vdash W}{W}$ where W is any closed formula, we will be no further ahead because the non theorems of arithmetic are not recursively enumerable. What this means is that there is no way in general that, given a W, we can establish that W is not a theorem even if W happens not to be a theorem. This in turn means that we are not even guaranteed that an arbitrary default rule of inference is effective i.e. there may be no algorithm which will inform us whether or not a given default rule of inference is applicable! From this we can conclude that the theories of a default theory may not be recursively enumerable. This situation is in marked contrast to what normally passes for a logic where, at the very least, the rules of inference must be effective and the theorems recursively enumerable.

Finally, it is not hard to see that default theories fail to satisfy the extension property [Hayes 1973] which all "respectable" logics do satisfy. (A logical calculus has the extension property iff whenever a formula is provable from a set P of premises, it is also provable from any set P' such that $P \subseteq P'$.)

[Kramosil 1975] attempts to establish some general results on default theories. Kramosil "proves" that for any such theory, the default rules are irrelevant in the sense that either the theory will be meaningless or the theorems of the theory will be precisely the same as those obtainable by ignoring the default rules of inference. Kramosil's result, if correct, would invalidate the

main point of this paper, namely that default theories play a prominent role in reasoning about the world. Fortunately, his "proof" relies on an incorrect definition of theoremhood so that the problem of characterizing the theorems of a default theory remain open.

5. CONCLUSIONS

Default reasoning may well be the rule, rather than the exception, in reasoning about the world since normally we must act in the presence of incomplete knowledge. Moreover, aside from mathematics and the physical sciences, most of what we know about the world has associated exceptions and caveats. Conventional logics, such as first order logic, lack the expressive power to adequately represent the knowledge required for reasoning by default. We gain this expressive power by introducing the default operator.

In order to provide an adequate formal (as opposed to heuristic) foundation for default reasoning we need a well articulated theory of default logic. This requires, in part, a theory of the semantics of default logic, a suitable notion of theoremhood and deduction, and conditions under which the default inference rules are effective and the set of theorems unique. Since in any realistic domain, all of the default schemata of Section 2 will be in force (together, no doubt, with others we have not considered) we require a deeper understanding of how these different schemata interact. Finally, there is an intriguing relationship between certain defaults and the complexity of the underlying representation. Both the closed world and frame defaults implicitly represent whole classes of first order axioms. Is this an accidental phenomemon or is some general principal involved?

ACKNOWLEDGEMENTS

This paper was written with the financial support of NRC grant A 7642. I am indebted to Brian Funt, Randy Goebel and Richard Rosenberg for their criticisms of an earlier draft of this paper.

REFERENCES

Bobrow, D.G. and Winograd, T., (1977). "An Overview of KRL-0, a Knowledge Representation Language," Cognitive Science, Vol.1, No.1, Jan. 1977.

Colmeraurer, A., (1973). Un System de Communication Home-Machine en Français, Rapport interne, UER de Luminy, Université d'Aix-Marseille, 1973.

Hayes, P.J., (1973). "The Frame Problem and Related Problems in Artificial Intelligence," in Artificial and Human Thinking, A. Elithorn and D. Jones (Eds.), Jossey-Bass Inc., San Francisco, 1973, pp.45-49.

Hayes, P.J., (1977). "In Defence of Logic," Proc. IJCAI-5, M.I.T., Cambridge, Mass., August 22-25, 1977, pp. 559-565.

Hewitt, C., (1972). Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot, A.I.Memo No. 251, M.I.T. Project MAC, Cambridge, Mass., April 1972.

Kanoui, H., (1976). "Some Aspects of Symbolic Integration via Predicate Logic Programming," SIGSAM Bulletin, 10, Nov. 1976, pp. 29-42.

Kramosil, I., (1975). "A Note on Deduction Rules with Negative Premises," Proc. IJCAI-4, Tbilisi, USSR, Sept. 3-8, 1975, pp. 53-56.

McCarthy J. and Hayes, P.J., (1969). "Some Philosophic Problems from the Standpoint of Artificial Intelligence," in Machine Intelligence 4, B. Meltzer and D. Michie (Eds.), Edinburgh University Press, Edinburgh, 1969, pp. 463-502.

Raphael, B., (1971). "The Frame Problem in Problem-Solving Systems," in Artificial Intelligence and Heuristic Programming, N.V. Findler and B. Meltzer (Eds.), Edinburgh University Press, Edinburgh.

Reiter, R., (1978). "On Closed World Data Bases," in Logic and Data Bases, H. Gallaire and J. Minker (Eds.), Plenum Press, New York, to appear.

Roberts, R.B. and Goldstein, I., (1977). The FRL Manual, A.I. Memo No. 409, M.I.T., Sept. 1977.

Roussel, P., (1975). PROLOG, Manuel de Reference et d'Utilisation, Group d'Intelligence Artificielle, U.E.R. de Marseille, France, 1975.

Sandewall, E., (1972). "An Approach to the Frame Problem, and its Implementation," in Machine Intelligence 7, B. Meltzer and D. Michie (Eds.), Edinburgh University Press, Edinburgh, pp. 195-204.

Sussman, G., Winograd, T., and Charniak,E., (1970). MICRO-PLANNER Reference Manual, A.I. MEMO No. 203, M.I.T., Cambridge, Mass., 1970.

Waldinger, R., (1975). Achieving Several Goals Simultaneously, Artificial Intelligence Center Technical Note 107, Stanford Research Institute, Menlo Park, Calif., July 1975.

Warren, D., (1974). WARPLAN: A System for Generating Plans, Memo No. 76, Dept. of Computational Logic, University of Edinburgh, June 1974.

Winograd, T., (1972). Understanding Natural Language, Academic Press, New York, 1972.

Woods, W.A., (1968). "Procedural Semantics for a Question-Answering Machine," AFIPS Conference Proceedings, Vol. 3, Part I, 1968, pp. 457-471.