A METHODOLOGY FOR USING A DEFAULT AND ABDUCTIVE REASONING SYSTEM

by

David Poole

Technical Report 89-20

September 18, 1989

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5 Canada

A methodology for using a default and abductive reasoning system

David Poole
Department of Computer Science,
University of British Columbia,
Vancouver, B.C., Canada, V6T 1W5
poole@cs.ubc.ca

September 18, 1989

Abstract

This paper investigates two different activities that involve making assumptions: predicting what one expects to be true and explaining observations. In a companion paper, a logic-based architecture for both prediction and explanation is proposed and an implementation is outlined. In this paper, we show how such a hypothetical reasoning system can be used to solve recognition, diagnostic and prediction problems. As part of this is the assumption that the default reasoner must be "programmed" to get the right answer and it is not just a matter of "stating what is true" and hoping the system will magically find the right answer. A number of distinctions have been found in practice to be important: between predicting whether something is expected to be true versus explaining why it is true; and between conventional defaults (assumptions as a communication convention), normality defaults (assumed for expediency) and conjectures (assumed only if there is evidence). The effects of these distinctions on recognition and prediction problems are presented. Examples from a running system are given.

1 Introduction

There have been many proposals for how to build nonmonotonic reasoning systems [Reiter80, McCarthy86, Moore85, Delgrande87]. There have, however, been very few discussions as to how one should *use* a non-monotonic reasoning system to solve the sorts of problems we want to solve (notable exceptions are plan recognition in [Kautz87], inheritance systems in [Etherington87] and the use of the abnormality predicate in [McCarthy86]). As even very weak logics can compute any computable function [Lloyd87, Theorem 9.6], there is nothing in principle that one of these can do that any other can't. The difference between each of these is in how they can be used to solve the sorts of problems that we want to solve. It is only by developing methodologies for using such systems that we will be able to compare and evaluate these systems.

This research follows from the conjecture that there is nothing wrong with classical logic in representing commonsense knowledge; there is, however, a problem with the assumption that to use logic we have to do deduction from our knowledge. We need to find different ways to use logic. As part of the Theorist project [PGA87, Poole88a], we are investigating how far we can get using a simple form of hypothetical reasoning, where the user provides a pool of possible hypotheses, instances of which can be used if consistent. We start with the hypothesis that "hypothetical reasoning, where the user provides the forms acceptable as hypotheses, and normal logic is used to test the consequence of our assumptions" is adequate for commonsense reasoning tasks. If this hypothesis is correct, we will have made a discovery in AI. Otherwise, by showing this hypothesis is not correct, we will have found examples where we need more power in our AI systems. This will also represent an advance in AI. To test (i.e., attempt to refute) this hypothesis, we need to build systems to solve real problems, which in particular, means we need to develop methodologies of how to solve problems. It may seem a bit weak to just develop programming methodologies, but this is, if you think about it, all that AI is doing; we are provided with a universal computing machine, we have to develop programming methodologies to make it suitable to solve problems and behave intelligently.

This work is done in the spirit of providing a very limited set of tools. Given these tools, we investigate how they can be used to solve problems. A repertoire of techniques can then be built to determine how to appropriately

use these tools. Only when these tools can be shown to be inadequate, or we have very good reasons why they should be augmented do we expand our set of tools. In this manner, the distinctions outlined in this paper were found from useful when using the system, explaining to others how to use the system and in building applications (see for example [Poole87]).

This paper is a companion paper to [Poole89b].

2 Distinctions

Example 2.1 Consider the following "knowledge":

> *A person may possibly have a brain tumour,*
> *a person may possibly have a broken leg,*
> *a brain tumour typically produces a headache, and*
> *a broken leg typically produces a sore leg and a bent leg.*

On the basis of this knowledge alone, if we observe that Randy has a bent leg, it is reasonable to hypothesise he may have a broken leg and the broken leg produced the bent leg. If we subsequently ask whether we predict a sore leg based on this information, we would say *yes*, as we hypothesise a broken leg which is typically sore. If we were asked whether we predict, on the evidence of a bent leg, that Randy has a headache we would say *no*, there is no reason to assume that he has a brain tumour given no evidence for it.

This simplistic example indicates a distinction between *explaining observations* and *predicting what we expect to be true.* There is also a distinction between *normality assumptions* (which we want to assume given no evidence to the contrary) and *abnormality assumptions* which we want to assume only if we have evidence.

Each of these distinctions is discussed in this section, and a system which respects such distinctions is outlined in the next section. Formal definitions, outlines of implementations and applications are discussed in later sections.

2.1 Prediction versus Explaining Observations

In example 2.1 we saw a distinction between predicting what we expected to be true as opposed to explaining actual observations.

These are both processes where we want to make assumptions in order to derive conclusions, but differ in the task they are carrying out. There are a number of differences imposed by the task:

1. There are some things which we only want to hypothesise if we have evidence. We don't want to hypothesise an invisible person in a picture or a rare disease in a patient if there is no evidence for them. However, if we have evidence of a rare disease, then we should hypothesise it. There are different hypotheses we bring to bear when asked whether we expect something is true or whether we are told that something is true and asked to find a plausible explanation of why it is true.

2. If we are trying to explain the observation g, it seems irrelevant that $\neg g$ is also able to be explained; this just means that in some other circumstances g is not true. If, however we are asked whether we predict g, it seems very relevant whether we can also explain its negation.

3. an observation is like a fact, in the sense that all of our theories must be consistent with it (in fact, in the proposed system the explanations imply the observations) whereas a prediction may or may not be explained or consistent with all future theories. We may find out that our predictions are incorrect, but our observations are given as correct.

This distinction between prediction and explaining observations is a difference in kind, not a difference in degree (this is important to avoid the question *why isn't there a continuum of values between them?*). Of course, we may observe some phenomena, and then make predictions based on that observation; this will be considered here as two activities, observing and then predicting.

The properties of each of these is discussed later; for now it is important to note the distinction.

2.2 Defaults and Conjectures

Example 2.1 shows a distinction between what I will call *defaults* (or "normality assumptions" which are assumed to be true, given no evidence to the contrary) and *conjectures* (or "abnormality assumptions") which are assumed only if we have evidence (for example, diseases or malfunctions in a system for diagnosis or prototypes in recognition or design tasks).

Defaults and conjectures are similar in that they are both statements that we can hypothesise, but differ in the task in which they can be hypothesised. Defaults are hypotheses which can be assumed for prediction. Conjectures are hypotheses that can be assumed for explaining observations. Because of this division of labour, there are a number of differences between them.

Defaults can be used unless there is reason to believe otherwise, for example, that some device is working correctly, that if you have broken your leg it is sore, that a bridge is passable. These are assumptions that can be used to predict something is true, unless there is evidence that they are incorrect. I also assume that they can be used to explain observations[1]. as I cannot think of an example where one would use them to predict something, but not use them to explain why something occurred (this is, however, not a crucial part of the theory).

This is contrasted with *conjectures* which one has up one's sleeve if one needs to explain some observation[2]. These may include such hypotheses as: someone has some disease, some device is malfunctioning in some way, or there is some object in a scene in a recognition task. Evidence is needed to assume these conjectures.

At first glance, this distinction seems to be more a difference in degree rather than a difference in kind (for example, in example 2.1 above, one could say that maybe Randy has a sore head because he may have a brain tumour which would cause a sore head). However, the distinction is the final essence is in the role that they each play. A hypothesis which can be used for prediction is a default, and one that can only be used for explaining observations is a conjecture.

2.3 Normality defaults and conventional defaults

We can distinguish two types of defaults:

- reasonable assumptions, which may be incorrect, but for the time being we will assume that they are true, a "normality default".

[1] I.e., they can also be used as conjectures.

[2] The user provides the system with formulae that can be used as conjectures if there is evidence for them. The term "conjecture" is used here to mean these formulae that the system has available to conjecture.

- communication conventions, where we know that something is true if we have no statement to the contrary, a "conventional default".

The classic AI example is that we have the default that birds fly, and know that Tweety is a bird, and know nothing else about Tweety, we conclude that Tweety flies. How one interprets this conclusion depends on whether the default is a normality default or a conventional default. If it is the former, the answer should mean that "we expect that Tweety flies, as birds typically do, but maybe she doesn't"; if the default was a conventional default, the answer should mean that "Tweety flies, as if she didn't fly you would have told us according to the convention that we have between us". I would claim that the second is still using default reasoning, but this distinction seems to be the distinction that Moore [Moore85] was making when he claimed that autoepistemic reasoning was not default reasoning.

This distinction is also important in solving the "multiple extension problem". Multiple extensions seem natural and to be expected for normality defaults, where if some individual is in two classes which normally have incompatible properties, it is to be expected that we can expect different conclusions based on the two classes. For conventional defaults, multiple extensions indicate a bug in our convention, as we have evidence that there is a consistent conclusion which we can draw which is incorrect (one of the extensions must be incorrect, as they all can't be correct as multiple extensions are always incompatible).

These defaults, at least for the tasks in this paper, seem to be *used* in the same way (this is supported by [Konolige87], where the formal equivalence between Default logic [Reiter80] and Autoepistemic Logic [Moore85] was proved). For the rest of this paper we will put both of these into one class called the *defaults*.

2.4 Facts and Hypotheses

One of the questions that arises when using a hypothetical reasoning system is when should some piece of knowledge be a fact and when should it be a hypothesis. The answer is that it is relative to the problem at hand. One person's facts may be another's hypotheses. This should not be seem as a bug in the theory, but as a feature.

The facts are those pieces of knowledge that for the sake of some argument

we are not prepared to give up. A default is some piece of knowledge we are prepared to give up if there is evidence to the contrary. In a similar way that our answers will be conditioned on the defaults used to conclude them, the set of explanations is conditioned by the meta-level assumptions made in building the knowledge base (these may or may not be explicit). Assumptions are made when building a knowledge base; if these are found to be wrong, we try to debug the knowledge base. This framework is the same theory formation and revision framework that the reasoning system itself uses.

One may often want to condition diagnoses with "assuming that the diseases are not acting pathologically and the problem is amongst the known diseases, the diagnosis is ...". If the symptoms cannot be explained, we know that this assumption is incorrect, and we can try to make explicit our assumptions to try to find out the correct diagnosis. This building of a new layer of the Theorist framework is not any different to the other tasks. In the rest of this paper, we assume that we are operating in one level of this hierarchy. See [Brewka89] for a discussion of multiple layers of hypotheses.

2.5 Facts and Observations

Perhaps a more difficult question is what knowledge should be added as facts and what should be added as observations. Facts and observations are both considered true of the domain under consideration, but they play very different roles as part of the framework. The answer "observations are those things we observe that need explaining" is a rather vacuous and unsatisfactory answer if there is no way to say what needs explaining and what does not.[3]

Instead I propose a convention that the facts consist of the general "background" knowledge about a domain, which includes physical and other constraints that we are not prepared to give up. The observations are all the things we observe about the particular case in hand. Thus facts can be seen as things necessarily true in the domain (as far as the designer is concerned), and the observations are the contingent facts, which happen to be true of the case under consideration. As far as the user is concerned, all she sees about a particular case are observations. The designer of the system can decide that

[3]Note that this is a problem of programming methodology, not of the theory or implementation of the system.

some observations can be treated as facts by writing them as conjectures (see section 4.3). This is perfectly consistent with the idea that conjectures are base causes that we can hypothesise if we have evidence.

One of the important properties of our system, is that once an observation has been explained, it is derivable in all resulting theories. As far as future questions and observations are concerned, the observation thus has the same status as a fact.

2.6 What this is not

This paper is not intended to present a theory of how one changes ones beliefs (i.e., how one changes from attending one theory of the world (scenario) to another). That seems to be either the role of a psychological theory (e.g., *How many scenarios do people consider at once? How many scenarios do people consider at all? How much evidence is required before someone changes their mind?* [Harman86]) or an implementation decision (e.g., *Should we build one theory at a time and undo relevant assumptions if we get into trouble?* [Doyle79] or *should we try to build all explanations or extensions at once?* [de Kleer86]). Both of these are very important issues but are not the subject of this paper.

This is intended to be a competence theory and not a performance theory or descriptive theory of nonmonotonic reasoning. This paper talks about consistency as something which can and should be checked in order to hypothesise something. It does not consider that people jump to conclusions with very little reasoning and only fix up their beliefs when they are convinced they are inconsistent, nor does it talk about how the processes can be done in real time. The psychological validity of this theory is not what is being considered here, nor are very efficient proof procedures.

Although this is presented in a theory formation framework, the proposed system is not intended to be a learning system. There is no way in this framework to generate new hypotheses. We are not trying to automatically generate general theories which are applicable to other cases.

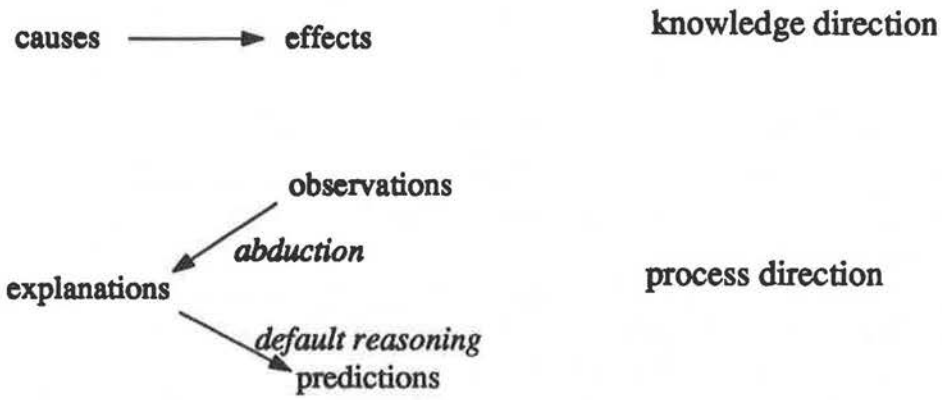

Figure 1: Architecture of the Theorist system presented here

3 A Default Reasoning System

3.1 Architecture

The architecture [Poole89b] we are considering is one where the system is provided with facts, defaults and conjectures. We assume that these provide the background knowledge about the domain being modelled (e.g., how diseases interact and how symptoms work in a diagnosis system, and general knowledge about objects, occlusion etc., in a recognition task), all specific knowledge about the particular case is added as observations (see section 4 for a description of the programming methodology).

The system is provided with a sequence of observations and abduces explanations of the observations. From each of these explanations we can ask what they predict. The system can also propose what observations it would like about the world in order to prune and refine its explanations.

The idea (figure 1) is that the user axiomatises the implication from causes to effects. When an observation is made, we abduce possible causes. From these causes, we predict what else we expect to be true. The same axiomatisation from causes to effects is used for both explanation and prediction.

See [Poole89b] for a more detailed description.

3.2 Theorist Framework

Theorist [PGA87, Poole88a, Poole89b] is a simple framework for hypo- deductive reasoning where the user provides the forms of the possibly hypotheses.

We assume that we are given a standard first order language over a countable alphabet [Enderton72]. By a formula we mean a well formed formula in this language. By an instance of a formula we mean a substitution of terms in this language for free variables in the formula.

Suppose A is a set of closed formulae and H is a set of (possibly open) formulae, a **scenario** of (A, H) is a set $A \cup D$ where D is a set of ground instances of elements of H such that $A \cup D$ is consistent. If g is a closed formula, an **explanation** of g from (A, H) is a scenario of (A, H) which implies g. An **extension** of (A, H) is the set of logical consequences of a maximal (with respect to set inclusion) scenario of (A, H).

[Poole88a] discusses how the above simple hypothetical reasoning framework can be used for default reasoning.

In accordance with the preceding discussion, the following sets of formulae are provided by the user:[4]

F is the set of *facts*, which we are taken as being true of the domain;

Δ is the set of *defaults*, possible hypotheses which can be used for prediction;

Π is the set of *conjectures*, possible hypotheses which can be used for explaining observations;

O is the set of *observations* that have been made about the actual world.

3.3 Explaining Observations

When explaining observations, we want to build a scenario as to *why* those observations could have occurred. This can be considered as abducing possible causes of the observations. We want to be able to hypothesise conjectures and defaults which would account for these observations.

[4] As far as the preceding semantics are given, the possible hypotheses, H, will in some cases be Δ and in some cases $\Pi \cup \Delta$; the given A will sometimes be F and sometimes an explanation of the observations.

Suppose we are given facts F, conjectures Π and defaults Δ, and O is observed. We want to explain O from $(F, \Pi \cup \Delta)$ (i.e., $\Pi \cup \Delta$ is the set of possible hypotheses). That is, we want sets P and D, instances of elements of Π and Δ respectively, such that

$F \cup P \cup D \models O$ and
$F \cup P \cup D$ is consistent

$P \cup D$ are the assumptions of the explanation.

See [Poole89b] for discussions on different ways to compare explanations and algorithms for computing explanations. We assume that we are computing the least presumptive (does not imply any other explanation) and minimal (contains not redundant hypotheses) explanations.

3.4 Prediction

When predicting what we expect to be true, the possible hypotheses we are prepared to use are the set Δ of defaults. The given formulae A will normally be an explanation of the observations. We want to predict some proposition g based on A and Δ if, assuming that everything that is not known to be acting "abnormally" is acting "normally", g is true.

Definition 3.1 *We* **predict** g *based on* (A, Δ) *if* g *is in every extension of* (A, Δ).

g is not in every extension of (A, Δ), if and only if there is some scenario S of (A, Δ), such that g is not able to be explained from (S, Δ) [Poole89b]. Based on our normality conditions and what we are given we cannot rule out S, and so we should not predict g. This is thus a very sceptical form of prediction.

For a more detailed discussion of alternative notions of prediction and how they can be implemented see [Poole89b].

3.5 Interacting with the system

When implementing Theorist we want a system in which we can add facts, defaults, etc., and then give observations and ask predictions based on what the system has been told.

The input language to the system is defined below.

fact w.
where w is a formula, means "$\forall w$"[5] is a fact.

default n.
where n is a name (predicate with only free variables as arguments) means n is a default[6].

conjecture n.
where n is a name means that n is a conjecture.

observe g.
where g is a closed formula, means that g is an observation. The result, $\mathcal{E}$ is the set of least presumptive and minimal explanations of all of the observations [Poole89b].

predict g, S.
where g is a formula and S is a scenario (usually one of the elements of $\mathcal{E}$), returns *yes* (together with the instance) if some instance of g is in every extension of S and *no* otherwise (not that we predict that g is false, but rather that we do not predict that it is true).

predict g.
where g is a formula returns *yes* (together with the instance) if some instance of g is in every extension of E, Δ for all $E \in \mathcal{E}$, and *no* otherwise.

4 Programming Methodology

It is not adequate to just define a representational language and leave it at that; it is also necessary to say how this language can be used to solve the sorts of problems we want to solve. This knowledge comes from experience with using the system. In this section we discuss some useful ways to use the system that we have found. I do not believe that one can or indeed should try to state knowledge without consideration as to how it is used

[5] $\forall w$ is the universal closure of w, that is, if w has free variables $\overline{v}$ then $\forall w$ means $\forall \overline{v}\ w$.

[6] This is not really a restriction on the forms of the defaults allowed. See [Poole88a] for a discussion on naming defaults.

(as, for example, [McDermott87] argued in the context of arguing against "logicism").

If we consider an algorithm as logic plus control [Kowalski79], logic programmers have realised that they must program both the logic and control to get their programs to run. Here we are considering how to program the logic; the discussion is independent of the control structure used.

4.1 Anticipating Explanations

Essentially statements that are to be used to predict what is true are added as facts if they are always true, or defaults if there are cases where they may not be applicable. What is always implied from a set defaults or conjectures (and thus can be used to rule out the hypotheses) are also added as facts. If a statement is possibly true, but can not be used for prediction, it is added as a conjecture.

Principle 1 *Any formula which can be used as part of a scenario, should be added as a fact if it is always true, as a default if it is used for prediction, or as a conjecture otherwise.*

For anything which could possibly be observed, one has to consider what an appropriate explanation would be. This may be the observation itself (see section 4.3) or more often, the observation is broken down into more primitive parts (causes) which in turn need to be explained. The implication of the observation from the causes can use any mixture of facts, defaults and conjectures. For example, if g is a potential observation and c is a potential cause of g, then c and $c \Rightarrow g$ could each be considered either as facts, defaults, conjectures or as observations which need to be explained. There is nothing in the formalism which forces us to think, for example, that $c \Rightarrow g$ should be a fact or default and c a conjecture.

Principle 2 *For each possible observation or prediction, consider what would be an appropriate explanation for it.*

4.2 Parametrizing Possible Hypotheses

When building systems using Theorist it is important to know how the way possible hypotheses can be parametrized to have different effects.

In general the free variables in possible hypotheses are the values on which the truth of the hypothesis depends. If, for example, the truth of a hypothesis depends on the time, then time should be a parameter of the possible hypothesis (then contradicting it at one time should not contradict it for other times). If the identity of some variable is irrelevant to the truth of a hypothesis, it should not be a parameter in the possible hypothesis.

Principle 3 *Parametrize possible hypotheses by those variables on which they depend.*

This is important because we want to actually imply the observations and predictions from the hypotheses.

Example 4.1 Consider the statement "you may assume that a person likes all dogs". This can be used to predict that some person likes some dog unless there is evidence to the contrary. If there is one dog which they do not like then we cannot assume that they like other dogs. This can be given by

default *likes-all-dogs*(P).
fact *likes-all-dogs*$(P) \wedge$ *person*$(P) \wedge$ *dog*$(D) \Rightarrow$ *likes*(P, D).

Making P and not D a parameter of the default means that the default is contradicted for a person if there is one dog they do not like.

For example, given also

fact *dog(fido)*.
fact *dog(honey)*.
fact *person(randy)*.
fact *person(sumo)*.
fact $\neg$*likes*$(randy, fido)$.

we can explain *likes*$(sumo, fido)$ but cannot explain *likes*$(randy, honey)$, as *likes-all-dogs*$(randy)$ is inconsistent with the facts.

This should be contrasted to the statement "you may assume that any person likes any dog". Here the existence of one dog that a person does not like should not prevent us from assuming they like other dogs. This can be specified by

default *likes-dog*(P, D).
fact *likes-dog*$(P, D) \wedge$ *person*$(P) \wedge$ *dog*$(D) \Rightarrow$ *likes*(P, D).

From this and the above facts, we can explain $likes(sumo, fido)$ and $likes(randy, honey)$ but not $likes(randy, fido)$

In contrast to other proposals where the hypotheses must be consistent with our observations (e.g., [Reiter87, de Kleer87]), our hypotheses must have the power to imply the observations. To do this the conjectures should be parametrized by the relevant inputs on which the cause depends as well as the possible outputs.

For example, if we want to consider malfunction d, that depends on parameters $I_1, ..., I_n$ (for example, incoming current, time of day, temperature in Antarctica) and predicts values for $O_1, ..., O_m$ (for example, temperature of a person, output current), the conjecture should be specified as being parametrized by all of these, namely as $hasmal_d(I_1, ..., I_n, O_1, ..., O_m)$. We are then allowed to hypothesise that the system has some outputs for the inputs given as

fact $\text{input}(I_1, ..., I_n) \wedge$
$hasmal_d(I_1, ..., I_n, O_1, ..., O_m) \wedge$
$reln(I_1, ..., I_n, O_1, ..., O_m)$
$\Rightarrow \text{output}(O_1, ..., O_m)$.
fact $c(I_1, ..., I_n, O_1, ..., O_m) \Rightarrow$
$\neg hasmal_d(I_1, ..., I_n, O_1, ..., O_m)$.

Where *reln* is some relation that must hold between the inputs and the outputs before we can use the hypothesis to predict the output from the given input, and c is some relation which cannot hold between the input and the output ($\neg c$ is a consequence of the malfunction). If we observe some output produced from some input, and if it fits the constraints of the malfunction (i.e. *reln* is true of them, and we cannot prove that c is true of them) then the appropriate instance of d can be conjectured as a cause of the output.

Example 4.2 Consider the domain of having a lamp connected to a battery. Suppose if a battery is acting normally its voltage is between 1.2 and 1.6; if it is overcharged, its voltage is above this, and if it is flat its voltage is below this range. The lamp will normally be lit if the voltage is over 1.3 and will be dim if the voltage is between 1.0 and 1.3, however if the voltage ever gets over 1.8 then the lamp will blow and never be normal again.

The following relations (with their intended interpretations) are used:

$battery(B)$ means B is a battery.

$lamp(L)$ means that L is a lamp.

$connect(B, L)$ means power supply B is connected to device L.

$voltage(B, V, T)$ means that at time T the voltage across battery B (and also across the lamp) is V volts.

$battOK(B, V, T)$ means that at time T, battery B is working OK and is producing V volts.

$overcharged(B, V, T)$ means that at time T, battery B is overcharged and is producing V volts.

$flat(B, V, T)$ means that at time T, battery B is flat and is producing V volts.

$lampOK(L, T)$ means that at time T, lamp L is working normally.

$dim(L, T)$ means that lamp L is dim at time T.

$lit(L, T)$ means that lamp L is lit at time T.

We can specify that the battery normally produces some voltage between 1.2 and 1.6 by

fact $battery(B) \wedge battOK(B, V, T) \Rightarrow voltage(B, V, T)$.
default $battOK(B, V, T)$.
fact $battOK(B, V, T) \Rightarrow 1.2 \leq V \wedge V \leq 1.6$.

We specify how the problems/malfunctions manifest themselves:

fact $battery(B) \wedge overcharged(B, V, T) \Rightarrow voltage(B, V, T)$.
conjecture $overcharged(B, V, T)$.
fact $overcharged(B, V, T) \Rightarrow V > 1.6$.
fact $battery(B) \wedge flat(B, V, T) \Rightarrow voltage(B, V, T)$.
conjecture $flat(B, V, T)$.
fact $flat(B, V, T) \Rightarrow V < 1.2$.

We also state that there cannot be two different voltages at any time (Note that this could have also be achieved by making *voltage* a function from time to the voltage at that time. It is added as a relation because we want to make it explicit when we are using the functionality of the relation.)

fact $voltage(B, V_1, T) \wedge voltage(B, V_2, T) \Rightarrow V_1 = V_2$.

Similarly we axiomatise how a lamp works normally:

fact $lamp(L) \wedge lampOK(L, T) \wedge voltage(L, V, T) \wedge V \geq 1.3 \Rightarrow lit(L, T)$.
fact $lamp(L) \wedge lampOK(L, T) \wedge voltage(L, V, T) \wedge 1.0 \leq V \wedge V < 1.3 \Rightarrow dim(L, T)$.
default $lampOK(L, T)$.
fact $lampOK(L, T) \wedge voltage(L, V, T) \Rightarrow V \leq 1.8$.
fact $\neg lampOK(L, T_0) \wedge before(T_0, T_1) \Rightarrow \neg lampOK(L, T_1)$.

We also say how lamps and batteries can be connected together,

fact $connect(B, L) \wedge voltage(B, T, V) \Rightarrow voltage(L, T, V)$.

Given no observations, we cannot predict that the voltage of some battery is any particular voltage, for example, 1.5 volts (as it is not true in all extensions), however we can predict

$$battery(B) \Rightarrow \exists V \; V \geq 1.2 \wedge V \leq 1.6 \wedge voltage(B, V, T)$$

for each B and T.

Given no observations, if we were asked to predict whether a lamp l connected to a battery b is lit at some time t, then the answer is *no*, as $\{battOK(b, 1.25, t)\}$ is a scenario from which $lit(l, t)$ cannot be explained. We can, however, predict $lit(t) \vee dim(t)$. There are infinitely many explanations of

$$battery(b) \wedge lamp(l) \wedge connect(b, l) \Rightarrow lit(t) \vee dim(t)$$

namely consisting of

$$\{battOK(b, V, t), lampOK(l, t)\}$$

for every V such that $1.2 \leq V \leq 1.6$. There is no scenario of (F, Δ) from which $lit(t) \vee dim(t)$ cannot be explained, given no observations.

Suppose we observe that when lamp l connected to battery b, it is dim at time t. This is specified as

observe $battery(b) \wedge lamp(l) \wedge connect(b,l) \Rightarrow dim(l,t)$.[7]

There are the least presumptive explanations:

$$\{battOK(b,V,t), lampOK(l,t)\}$$

for $1.2 \leq V \leq 1.3$ and

$$\{flat(b,V,t), lampOK(l,t)\}$$

for $1.0 \leq V < 1.2$ (only the former are minimal abnormality explanations [Poole89b]). We only predict things which are true in all extensions of these explanations.

If we observe that the voltage is 1.25 at time t_0,

observe $battery(b) \Rightarrow voltage(b, 1.25, t_0)$

there is one explanation, namely

$$\{battOK(b, 1.25, t_0), lampOK(t_0)\}$$

Example 4.3 Consider the hunting, robbing example of [Kautz87]. Suppose that if someone is hunting, they get a gun and go to a forest. The instance of hunting we are considering depends on the agent, the gun and the forest. We can then write this as

conjecture $hunting(A,G,F)$.
fact $hunting(A,G,F) \Rightarrow get(A,G) \wedge goto(A,F)$.

$hunting(A,G,F)$ means that agent A is hunting in forest F with gun G. Note that knowing a person is hunting does not imply that he gets some particular gun. However, by parametrizing the hypothesis on what it depends, we can explain why a person went to a particular forest, but the knowledge that a person went hunting only implies that they went to some forest. For example, if we observe that Henry went to Sherwood forest:

[7]Note that the observation is an implication. We need to explain that when b is a battery and l is a lamp, and they are connected l is dim. We do not want to explain why b is a battery, or why they are connected together. Putting these on the left hand side of the implication is like adding them as facts for the purpose of the observation.

observe $goto(henry, sherwood\_forest)$

there is the explanation

$$\{hunting(henry, G, sherwood\_forest)\}$$

for each ground instance of G. All we can predict is that there exists a gun that Henry gets; each explanation logically implies that Fred goes to Sherwood Forest, and each implies he gets some gun.

4.3 Observations and Facts

In section 2.5 it was claimed that all of the generalised knowledge about a domain should be added as facts and all knowledge about a particular case should be added as observations. This convention makes a clear distinction for the user of the system, but requires the builder of the knowledge base to be aware of this. However, if one follows principle 2, the conjectures that the designer provides should include all of those possible observations that one wants to treat as facts. This is entirely within the spirit of conjectures; we just don't want a deeper analysis of the cause of these observations.

For example, if we want to allow the age of a patient to be added as an observation, but do not want a deep analysis of why this is the observed age, then we can add

conjecture $age(P, A)$.

If we find out the age of Jen, this is added as

observe $age(jen, 25)$.

There is one least presumptive explanation:

$$\{age(jen, 25)\}.$$

This has the same effect as adding $age(jen, 25)$ as a fact as the least presumptive explanations will always contain $age(jen, 25)$.

The conjectures are thus whatever we are prepared to accept as components in explanations for observations whether they are formulae that don't really need to be explained or are deep causes for complex behaviour.

4.4 Causes and Symptoms

One of the ways of looking at recognition and diagnostic tasks is to find the causes of symptoms [Cox87]. There are cases when something can be considered a cause sometimes and symptom at other times. If not handled appropriately, this may become a problem if we prefer the least presumptive explanation (see section 3.3). Appropriate structuring of the knowledge base will avoid these problems. Consider the following example:

Example 4.4 Suppose we want to represent the sentences

> *Sometimes people sneeze because they have a cold.*
> *Sometimes people just sneeze.*

One representation of this may be

> **conjecture** *sneezes*(X).
> **conjecture** *has-cold*(X).
> **default** *sneezing-because-of-cold*(X).
> **fact** *sneezing-because-of-cold*$(X) \wedge$ *has-cold*$(X) \Rightarrow$ *sneezes*(X).

If we observe

> **observe** *sneezes*(*eric*).

there is one least presumptive explanation, namely

$$\{sneezes(eric)\}$$

The explanation that eric has a cold is not considered because it is more presumptive than the other explanation. There may be a problem here with interpretation; we should not consider this answer as meaning the second sentence above (i.e. that he is just sneezing for no reason). This answer means that he is sneezing, and that is considered as a cause in itself. It does not exclude that he is sneezing because of a cold.

If, however, we want to distinguish between the two causes then the appropriate way to represent this is

> **conjecture** *random-irritation*(X).
> **conjecture** *has-cold*(X).
> **default** *sneezing-because-of-cold*(X).

fact *sneezing-because-of-cold*$(X) \wedge$*has-cold*$(X) \Rightarrow$*sneezes*(X).
default *just-sneezing*(X).
fact *just-sneezing*$(X) \wedge$*random-irritation*$(X) \Rightarrow$ *sneezes*(X).

In this case there are two least presumptive explanations of eric sneezing:

$$\{\textit{random-irritation}(\textit{eric}),\ \textit{just-sneezing}(\textit{eric})\}$$

$$\{\textit{has-cold}(\textit{eric}),\ \textit{sneezing-because-of-cold}(\textit{eric})\}$$

Here we can distinguish the different causes.

4.5 Hypo-deductive reasoning

What has been presented so far can be seen as one instance of, what Hempel called *deductive-nomological explanations* [Hempel66, Popper62, Quine78]. McDermott [McDermott87] has criticised this as a method for AI reasoning, so it is interesting to consider how Theorist copes with the problems in Hempel's formalism.

Deductive-nomological explanations of E are "deductive arguments whose conclusion is the explanandum sentence E, and whose premiss-set, the explanans, consists of general laws and of other statements which make assertions about particular facts" [Hempel66, p. 51].

In Theorist, the problem of what are "general laws" are avoided by the acceptable hypotheses being part of the background knowledge as much as other facts about the world. Thus, we are not doing arbitrary theory formation, where we may explain "Elizabeth is the Queen of Australia" by the law "Copper conducts electricity" and the true statement, "Elizabeth is the Queen of Australia or copper doesn't conduct electricity".

Notice that Hempel shows that observation, "must be true", given that other things are true. The laws are things which are always true. Thus he is trying to explain why some observation must have occurred, given all of the other knowledge in the system. The Theorist hypotheses, are rather explanations which may occur. There is no notion of likelihood, only possibility (the probability of an explanation is an orthogonal issue [Neufeld87a]).

Thus there is no problem arising for the example of Selma McGillicuddy of Seraucus who won the New Jersey lottery for the second time in the last two months [McDermott87, p. 153]. If we allow chance as a reasonable explanation for someone winning the lottery we can write

conjecture $wins\_lottery\_by\_chance(P,T)$.
fact $wins\_lottery\_by\_chance(P,T) \Rightarrow wins\_lottery(P,T)$.

If we observe that Selma won the lottery on February 13 and March 3, we have the explanation

$$\{wins\_lottery\_by\_chance(selma, feb\_13),\\ wins\_lottery\_by\_chance(selma, march\_3)\}$$

This does indeed imply the observations.

Notice that we have just followed the programming methodology above, and not made any special allowance for this example. Note also that we still do not predict that Selma wins the lottery given no evidence, as even if above were defaults and not conjectures, we only predict what is in all extensions, and in a complete axiomatisation there would be a scenarion where someone else wins the lottery.

4.6 Abstraction and Causal Hierarchies

[Kautz87] dismisses hypo-deductive reasoning as being unsuitable for plan recognition because it does not work properly for the combination of causal implication and abstraction implication.

For example, consider that if someone is going hunting, they get a gun and go into the woods. Suppose we also know that a MIG machine gun is a sort of gun. If we want to answer the question "why did Henry get the MIG?", then [Kautz87] claims that the answer to the question is "to go hunting; as the mig is a sort of gun and and a gun can be used to go hunting". He does not also consider the meaning of the sentence as "why did he get a MIG as opposed to another sort of gun?". In this section, I show how the above methodology naturally solves the problems.

Firstly, we define the abstraction hierarchies in the normal manner, namely by making them implications, to say that a machine gun is a type of gun, that a hunting activity is a physical activity, and that a forest is a wild place, we say

fact $machine\_gun(X) \Rightarrow gun(X)$.
fact $hunting\_activity(E) \Rightarrow physical\_activity(E)$.
fact $forest(P) \Rightarrow wild(P)$.

If we do not need an explanation as to why the gun was a machine gun (i.e., the fact that it is a machine gun is an adequate explanation of it being a machine gun), then we say

conjecture $machine\_gun(X)$.
conjecture $gun(X)$.
conjecture $hunting\_activity(E)$.
conjecture $forest(P)$.
conjecture $wild(P)$.

The standard way to add type information (which is the sort of information in an abstraction hierarchy), is to form an implication of a type predicate (parametrized by the variable) implying the sentence in which the variable appears.

We can use this method then to express our knowledge about types. For example to say that wanting exercise is a reasonable base cause for doing a physical activity, we can say:

default $doforexercise(P, E)$.
fact $doforexercise(P, E) \wedge physical\_activity(E) \wedge wants\_exercise(P, E) \Rightarrow do(P, E)$.
conjecture $wants\_exercise(P, E)$.

The only "trick" we have done to represent this is to use the standard trick (principle 3) of parametrizing the conjectures over their possible values.

To say that when any agent A goes hunting they get a gun and go to a wild area, we say

default $gohunting(A, W, P)$.
fact $gohunting(A, W, P) \wedge hunting\_activity(h(W, P)) \wedge gun(W) \wedge wild(P) \wedge do(A, h(W, P)) \Rightarrow get(A, W) \wedge goto(A, P)$.

If we observe fred getting a machine gun, we name the machine gun, and write:

observe $get(fred, mig) \wedge machine\_gun(mig)$.

The least presumptive explanations of this contain the ground instances of $wants\_exercise(fred, h(mig, A1))$ and $wild(A1)$.

The importance of this example is that not only can we use abstraction and causal hierarchies in the Theorist system, but can develop the program by following the preceding programming methodology and standard techniques.

5 Applications

5.1 Diagnosis

In this section we show how the preceding outline can be a basis for formalising model-based diagnosis. This theory, as a theory of diagnosis, is an attempt to bridge the gap between diagnosis from first principles [Reiter87, Davis84, Genesereth84, de Kleer87], and more experience-based diagnosis based on knowledge as to how diseases and malfunctions normally manifest themselves [Weiss78, Patil81, Popl83, Brown82]. See [Poole88b, Poole89a] for detailed comparisons of various approaches.

In diagnosis from first principles, one has a model of the intended behaviour of the system. Any discrepancy between the predicted and observed behaviour means that the assumptions that components are working correctly is inconsistent with the observations, and so we can prove that some components are not working correctly. Reiter [Reiter87] defines a diagnosis as a minimal set of assumptions that components are faulty, together with the assumption that all other components are working correctly that is consistent with all observations of the system. That is, there are defaults of normality, and a diagnosis corresponds to an extension. He is doing the predictive half of the architecture presented here, and does not do any abduction.

When doing a diagnosis, we want to find out what is wrong with some system. One way to do this is to find some minimal set of components we need to assume are faulty given our evidence. Somehow we have to make these assumptions relevant to the observations and not to always say that we should just assume nothing. Reiter minimises abnormality assumptions and maximises normality assumptions so that the observations are consistent. In the framework suggested in this paper, we minimise all assumptions, however to stop always degenerating to the case of making no assumptions, we must have our assumptions implying the actual observations.

The main consequence of this distinction is that we have the ability as well as the obligation to state how problems manifest themselves. We must not only state how normal components act, but also how abnormal components act. This is not as big an imposition as it may seem as we can always say that a component is abnormal if it is working in some way that is different to what was designed.

Example 5.1 (Genesereth and Reiter) This example is derived from [Genesereth84, Fig. 8, p416] and [Reiter87, Example 2.2, p. 60]. To specify the intended action of an and-gate, Reiter gives the axiom (here we have modified Reiter's notation slightly to allow multiple observations as in [Genesereth84])

$$andg(X) \wedge \neg ab(X) \Rightarrow out(X,T) = and(in1(X,T), in2(X,T))$$

This axiom tells us what happens if a gate is working normally. It does not tell us what happens if the gate is acting abnormally. By abnormal, Reiter means that there exists some input value for which it gives the incorrect output value.

In Theorist, we need to parametrize the assumption and talk about acting normally for some inputs and acting abnormally for other inputs. If we decide that the relevant parameters to the normality assumption are the inputs to the gate (i.e., not on the time of day[8], or the amount of money in my bank account), then we use the relations $ab(X, I_1, I_2, O)$ which means that gate X is working abnormally for inputs I_1 and I_2, and producing output O, as well as the corresponding $ok(X, I_1, I_2, O)$. The operations of the gate can then be specified as

fact $andg(X) \wedge ok(X, in1(X), in2(X), out(X))$
$\Rightarrow out(X,T) = and(in1(X,T), in2(X,T))$.
default $ok(X, I1, I2)$.
fact $andg(X) \wedge ab(X, in1(X), in2(X), V)$
$\wedge V \neq and(in1(X,T), in2(X,T))$
$\Rightarrow out(X,T) = V$.
conjecture $ab(X, I1, I2, O)$.

The first fact says that the output of a normal and-gate is the conjunction of the inputs. The second fact says that the output of an abnormal and-gate (i.e. abnormal for the particular input values) is some value which is different to the conjunction of the inputs.

The main differences between the diagnoses is that Theorist does not need to make assumptions about parts which are not relevant to the diagnosis (we

[8] By not making the value depend on the time, we are making the non-intermittency assumption, namely that the value of the outputs of a gate depends only on the inputs and not on the time. If we did not want to make this assumption, we could add T as a parameter to our assumptions.

minimise all assumptions, whereas Reiter maximises normality assumptions). By *ok* we mean that the gate is working normally for the particular inputs being considered. Reiter means (by $\neg ab(X)$) that X is working normally for all inputs. Theorist can have a gate being OK for some inputs and not OK for other inputs.

It is straight forward to incorporate fault models into Theorist. For example, to say that faulty gates are either stuck at one or stuck at zero we replace the *ab* conjecture with more specific knowledge:

fact $andg(X) \wedge stuck(X,V) \Rightarrow out(X,T) = V.$
conjecture $stuck(X,V).$

Reiter would specify this as

$andg(X) \wedge ab(X) \wedge \neg ab'(X) \Rightarrow stuck1(X) \vee stuck0(X)$
$stuck1(X) \Rightarrow out(X,T) = 1$
$stuck0(X) \Rightarrow out(X,T) = 0$

Note that the use of this axiom is very different to the use of the Theorist version. This is only used to say that a gate by default is not broken because it is not stuck at one or stuck at zero. This is only useful if we can indeed prove that one of these is not the case. For more complicated cases it is easy to imagine a situation where we cannot actually prove that some abnormality does not occur. This is very different to being able to conjecture a fault. Also Reiter's diagnosis does not say that the gate is stuck at one, it just says that the gate is abnormal.

There seems to be no way to prove that one model of diagnosis is better than another, except by using each for a number of different applications. See [Poole88b, Poole89a] for a more detailed comparison of the models of diagnosis.

5.2 Recognition

As a final domain consider the framework for depiction and image interpretation of Reiter and Mackworth [Reiter89].

One Theorist representation of this would consist of having the possible scene objects as conjectures. These conjectures are the building blocks of the scene domain. We hypothesise a scene that could produce the image rather

than proving what the scene must be like. In this implementation they are parametrized by the object in the image that they denote. Thus, for example, $land(X)$ means that image object X represents land in the scene.

One difference between the implementation here and Reiter and Mackworth's is that we need to name the ends of the open chains. One end of a linear scene object is arbitrarily given the name 0 and the other is assigned the value 1. This allows us to identify the ends of the chains that correspond to mouths and sources of rivers for example. Thus, for example, $joins(X, Y, E)$ means that end E of road X joins scene object Y. Similarly $mouth(R, L, E)$ means that end E of river R joins scene object L, and $source(R, E)$ means that end E of linear scene object R is a place where is starts, not joining anything.

The following are the conjectures of possible scene objects that are the building blocks of scene descriptions.

conjecture $land(X)$.
conjecture $water(X)$.
conjecture $road(X)$.
conjecture $river(X)$.
conjecture $shore(X)$.

conjecture $joins(X, Y, E)$.
conjecture $cross(X, Y)$.
conjecture $beside(X, Y)$.
conjecture $mouth(R, L, E)$.
conjecture $source(R, X)$.
conjecture $loop(X)$.
conjecture $inside(X, Y)$.
conjecture $outside(X, Y)$.

Next we need to axiomatise the relationship between image objects and scene objects. These correspond to Reiter and Mackworth's relational mappings. Note that we need only axiomatise the implication from scene to image and not the equivalence between scene and image descriptions.[9]

[9]Note that here we have not used the depicts relation (the Δ of Reiter and Mackworth),

fact $beside(X,Y) \Rightarrow bounds(X,Y).$
fact $joins(X,Y,E) \Rightarrow tee(X,Y,E).$
fact $mouth(X,Y,E) \Rightarrow tee(X,Y,E).$
fact $cross(X,Y) \Rightarrow chi(X,Y).$
fact $source(X,E) \Rightarrow open(X,E).$
fact $loop(X) \Rightarrow closed(X).$
fact $inside(X,Y) \Rightarrow interior(X,Y).$
fact $outside(X,Y) \Rightarrow exterior(X,Y).$
fact $area(X) \Rightarrow region(X).$
fact $linear(X) \Rightarrow chain(X).$

We can also have axioms that allow for a taxonomy of descriptions in the scene domain.

fact $land(X) \vee water(X) \Rightarrow area(X).$
fact $river(X) \vee road(X) or shore(X) \Rightarrow linear(X).$

Finally we need to specify constraints on the scene objects.

fact $river(X) \wedge river(Y) \Rightarrow \neg cross(X,Y).$
fact $shore(X) \vee shore(Y) \Rightarrow \neg cross(X,Y).$
fact $river(R) \wedge mouth(L1,R,0) \Rightarrow \neg mouth(L2,R,1).$
fact $river(R) \wedge road(Y) \wedge joins(R,Y,N) \Rightarrow start(R,N).$
fact $source(X,Y) \Rightarrow start(X,Y).$
fact $river(R) \wedge start(R,0) \Rightarrow \neg start(R,1).$
fact $river(R) \wedge (river(L) \vee shore(L)) \Rightarrow \neg joins(R,L,N).$
fact $road(X) \vee road(Y) \Rightarrow \neg mouth(X,Y,N).$
fact $shore(X) \Rightarrow \neg source(X,N) \wedge \neg joins(X,A,N).$
fact $shore(X) \wedge river(A) \Rightarrow \neg joins(A,X,N).$
fact $river(X) \Rightarrow \neg loop(X).$
fact $shore(X) \wedge inside(X,Y) \wedge outside(X,Z) \Rightarrow (land(Y) \Rightarrow$

---

to give a relationship between scene and image objects. This can be incorporated into this axiomatisation by adding the relation $depicts(Xi,Xs)$ (meaning image object Xi depicts scene object Xs) to join the variables in the left and right hand side of the implications. There can either be a conjecture $depicts(I,S)$ to enable the relationship between scene and image objects to be part of the explanation, or there can be an axiom that says that for every image object there is a scene object to which it corresponds (thus allowing Skolemisation to give a name to the scene object corresponding to each image object). This was not done in order to keep the axiomatisation as simple as possible.

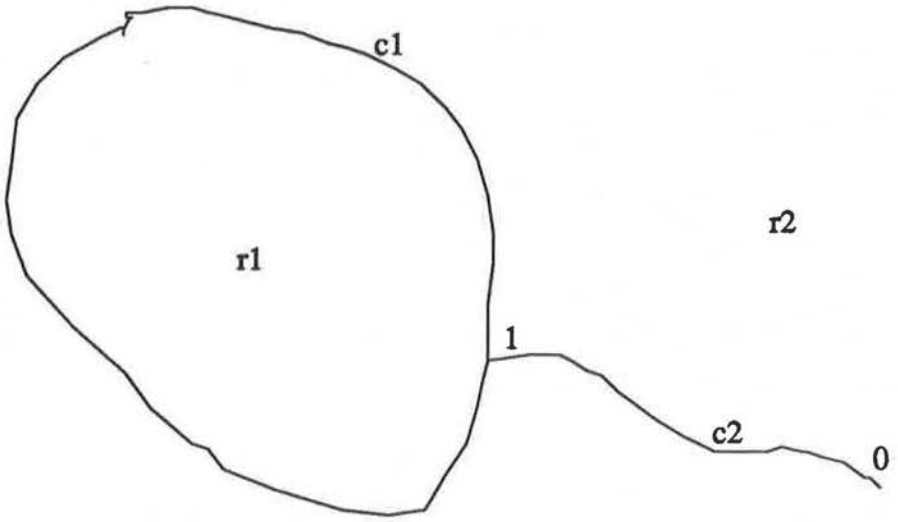

Figure 2: Simple sketch-map image

$\neg land(Z)) \wedge (water(Z) \Rightarrow \neg water(Y))$.
fact $beside(X,Y) \wedge (road(X) \vee river(X)) \Rightarrow \neg water(Y)$.

The image description is given as an observation. Consider the sketch map of figure 2. The corresponding observation is:

observe $chain(c1) \wedge chain(c2) \wedge region(r1) \wedge region(r2) \wedge tee(c2, c1, 1) \wedge bounds(c2, r2) \wedge bounds(c1, r1) \wedge bounds(c1, r2) \wedge interior(c1, r1) \wedge exterior(c1, r2), open(c2, 0) \wedge closed(c1)$.

The corresponding minimal explanations are

1. $\{loop(c1)$, $source(c2, 0)$, $outside(c1, r2)$, $inside(c1, r1)$, $beside(c1, r2)$, $beside(c1, r1)$, $beside(c2, r2)$, $joins(c2, c1, 1)$, $land(r2)$, $land(r1)$, $road(c2)$, $road(c1)\}$

2. $\{loop(c1)$, $source(c2, 0)$, $outside(c1, r2)$, $inside(c1, r1)$, $beside(c1, r2)$, $beside(c1, r1)$, $beside(c2, r2)$, $mouth(c2, c1, 1)$, $land(r2)$, $water(r1)$, $river(c2)$, $shore(c1)\}$

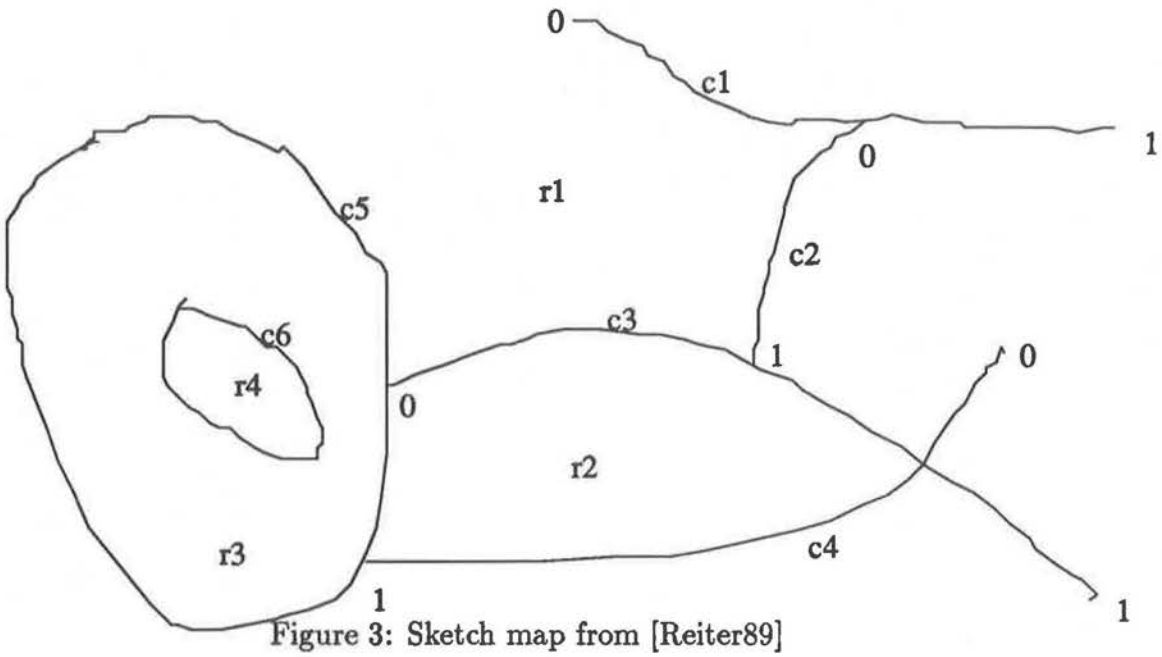

Figure 3: Sketch map from [Reiter89]

3. $\{loop(c1)$, $source(c2,0)$, $outside(c1,r2)$, $inside(c1,r1)$, $beside(c1,r2)$, $beside(c1,r1)$, $beside(c2,r2)$, $joins(c2,c1,1)$, $land(r2)$, $water(r1)$, $road(c2)$, $shore(c1)\}$

The explanations are thus detailed descriptions of the scene that imply the image observed.

As a more complicated example, consider the example of [Reiter89] given in figure 3. The observation corresponding to the image is

observe $chain(c1) \wedge open(c1,0) \wedge open(c1,1) \wedge region(r1) \wedge bounds(c1,r1) \wedge chain(c2) \wedge tee(c2,c1,0) \wedge bounds(c2,r1) \wedge chain(c3) \wedge open(c3,1) \wedge tee(c2,c3,1) \wedge bounds(c3,r1) \wedge region(r2) \wedge bounds(c3,r2) \wedge chain(c4) \wedge open(c4,0) \wedge chi(c3,c4) \wedge chi(c4,c3) \wedge bounds(c4,r1) \wedge bounds(c4,r2) \wedge chain(c5) \wedge closed(c5) \wedge tee(c4,c5,1) \wedge tee(c3,c5,0) \wedge bounds(c5,r1) \wedge bounds(c5,r2) \wedge exterior(c5,r1) \wedge exterior(c5,r2) \wedge region(r3) \wedge bounds(c5,r3) \wedge interior(c5,r3) \wedge chain(c6) \wedge closed(c6) \wedge bounds(c6,r3) \wedge region(r4) \wedge bounds(c6,r4) \wedge interior(c6,r4) \wedge exterior(c6,r3)$

The corresponding explanations are:

1. {*outside*(c6, r3), *inside*(c6, r4), *beside*(c6, r4), *land*(r4), *beside*(c6, r3), *loop*(c6), *shore*(c6), *inside*(c5, r3), *beside*(c5, r3), *water*(r3), *outside*(c5, r2), *outside*(c5, r1), *beside*(c5, r2), *beside*(c5, r1), *mouth*(c3, c5, 0), *joins*(c4, c5, 1), *loop*(c5), *shore*(c5), *beside*(c4, r2), *beside*(c4, r1), *cross*(c4, c3), *cross*(c3, c4), *source*(c4, 0), *road*(c4), *beside*(c3, r2), *land*(r2), *beside*(c3, r1), *mouth*(c2, c3, 1), *source*(c3, 1), *river*(c3), *beside*(c2, r1), *joins*(c2, c1, 0), *river*(c2), *beside*(c1, r1), *land*(r1), *source*(c1, 1), *source*(c1, 0), *road*(c1)}

2. {*outside*(c6, r3), *inside*(c6, r4), *beside*(c6, r4), *land*(r4), *beside*(c6, r3), *loop*(c6), *shore*(c6), *inside*(c5, r3), *beside*(c5, r3), *water*(r3), *outside*(c5, r2), *outside*(c5, r1), *beside*(c5, r2), *beside*(c5, r1), *mouth*(c3, c5, 0), *joins*(c4, c5, 1), *loop*(c5), *shore*(c5), *beside*(c4, r2), *beside*(c4, r1), *cross*(c4, c3), *cross*(c3, c4), *source*(c4, 0), *road*(c4), *beside*(c3, r2), *land*(r2), *beside*(c3, r1), *joins*(c2, c3, 1), *source*(c3, 1), *river*(c3), *beside*(c2, r1), *joins*(c2, c1, 0), *road*(c2), *beside*(c1, r1), *land*(r1), *source*(c1, 1), *source*(c1, 0), *road*(c1)]}

3. {*outside*(c6, r3), *inside*(c6, r4), *beside*(c6, r4), *land*(r4), *beside*(c6, r3), *loop*(c6), *shore*(c6), *inside*(c5, r3), *beside*(c5, r3), *water*(r3), *outside*(c5, r2), *outside*(c5, r1), *beside*(c5, r2), *beside*(c5, r1), *joins*(c3, c5, 0), *mouth*(c4, c5, 1), *loop*(c5), *shore*(c5), *beside*(c4, r2), *beside*(c4, r1), *cross*(c4, c3), *cross*(c3, c4), *source*(c4, 0), *river*(c4), *beside*(c3, r2), *land*(r2), *beside*(c3, r1), *joins*(c2, c3, 1), *source*(c3, 1), *road*(c3), *beside*(c2, r1), *joins*(c2, c1, 0), *road*(c2), *beside*(c1, r1), *land*(r1), *source*(c1, 1), *source*(c1, 0), *road*(c1)}

4. {*outside*(c6, r3), *inside*(c6, r4), *beside*(c6, r4), *land*(r4), *beside*(c6, r3), *loop*(c6), *road*(c6), *inside*(c5, r3), *beside*(c5, r3), *land*(r3), *outside*(c5, r2), *outside*(c5, r1), *beside*(c5, r2), *beside*(c5, r1), *joins*(c3, c5, 0), *joins*(c4, c5, 1), *loop*(c5), *road*(c5), *beside*(c4, r2), *beside*(c4, r1), *cross*(c4, c3), *cross*(c3, c4), *source*(c4, 0), *road*(c4), *beside*(c3, r2), *land*(r2), *beside*(c3, r1), *joins*(c2, c3, 1), *source*(c3, 1), *road*(c3), *beside*(c2, r1), *joins*(c2, c1, 0), *road*(c2), *beside*(c1, r1), *land*(r1), *source*(c1, 1), *source*(c1, 0), *road*(c1)}

5. {*outside*(c6, r3), *inside*(c6, r4), *beside*(c6, r4), *water*(r4), *beside*(c6, r3), *loop*(c6), *shore*(c6), *inside*(c5, r3), *beside*(c5, r3), *land*(r3), *outside*(c5, r2), *outside*(c5, r1), *beside*(c5, r2), *beside*(c5, r1), *joins*(c3, c5, 0), *joins*(c4, c5, 1), *loop*(c5), *road*(c5), *beside*(c4, r2), *beside*(c4, r1), *cross*(c4, c3), *cross*(c3, c4), *source*(c4, 0), *road*(c4), *beside*(c3, r2), *land*(r2), *beside*(c3, r1), *joins*(c2, c3, 1),

$source(c3,1), road(c3), beside(c2,r1), joins(c2,c1,0), road(c2), beside(c1,r1), land(r1), source(c1,1), source(c1,0), road(c1)\}$

6. $\{outside(c6,r3), inside(c6,r4), beside(c6,r4), land(r4), beside(c6,r3), loop(c6), shore(c6), inside(c5,r3), beside(c5,r3), water(r3), outside(c5,r2), outside(c5,r1), beside(c5,r2), beside(c5,r1), joins(c3,c5,0), joins(c4,c5,1), loop(c5), shore(c5), beside(c4,r2), beside(c4,r1), cross(c4,c3), cross(c3,c4), source(c4,0), road(c4), beside(c3,r2), land(r2), beside(c3,r1), joins(c2,c3,1), source(c3,1), road(c3), beside(c2,r1), joins(c2,c1,0), road(c2), beside(c1,r1), land(r1), source(c1,1), source(c1,0), road(c1)]\}$

These correspond exactly to the interpretations of [Reiter89].

The notable difference between this axiomatisation and the axiomatisation of Reiter and Mackworth is in what we do not have to specify. We do not have to have explicit domain closure axioms, nor do we require a unique names assumption. Rather than requiring both implications from the scene to image and also the implications from image to scene, we require only the "graphics" implication from scene to image. It is expected that this information will be more stable and more available than the inverse implication.

One other notable difference is that we automatically solve a "frame" problem. If we do not observe an end to a chain in the image (for example because the chain runs off the side of the image), we do not have to specify how that chain ends, or whether it is a loop or not.

Which of the formulations is better able to be expanded into more complicated and realistic domains is an open question.

6 Conclusion

This paper presents arguments why some distinctions are important in hypothetical reasoning, discusses a system which uses these distinctions and demonstrates ways in which the resulting system can be used to solve commonsense reasoning tasks.

Those of us building non-monotonic reasoning systems will eventually have to say how they can be used to solve real problems. To show they are useful for AI we have to develop appropriate programming methodologies. As they are usually very powerful logics, that can express any computable function, the only way that we can make an empirical statement that says

"this system is a good representation system" is to show how it can be used as a representation system. I do not believe that we will make advances unless we are explicit about how to use these systems. We cannot expect to just throw "correct" knowledge at them and get appropriate answers back.

This paper is part of the endeavour to test the hypothesis that hypothetical reasoning with normal logic is powerful enough to characterise commonsense reasoning [Poole88a, Poole89b]. The results from this are particularly encouraging, for example in showing that the problems with abduction in [McDermott87] are not really problems at all.

This paper is orthogonal to the issues of comparing explanations and scenarios [Poole85, Neufeld87a, Goebel88], and of building efficient implementations [Poole89b, PGA87].

There is still much more work that needs to be done, particularly in determining how to axiomatise a domain. We have only scratched the surface here.

Acknowledgements

This work could not have been done without the ideas, criticism, feedback and support of Randy Goebel, Eric Neufeld, Paul Van Arragon, Romas Aleliunas and Scott Goodwin. Thanks to Alan Mackworth, Andrew Csinger and Alex Kean for valuable comments on this paper. This research is supported under NSERC grant OGPOO44121.

References

[Brewka89] G. Brewka, "Preferred subtheories: an extended logical framework for default reasoning", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, pp. 1043-1048.

[Brown82] J. S. Brown, R. R. Burton and J. de Kleer, "Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III", in D. Sleeman and J. S. Brown (ed.), *Intelligent Tutoring Systems*, Academic Press, New York, pp. 227-282.

[Cox87] P. T. Cox and T. Pietrzykowski, *General Diagnosis by Abductive Inference*, Technical report CS8701, School of Computer Science, Technical University of Nova Scotia, April 1987.

[Davis84] R. Davis, "Diagnostic Reasoning Based on Structure and Behaviour", *Artificial Intelligence 24*, pp. 347-410.

[Delgrande87] J. P. Delgrande, "A First-Order Conditional Logic for Prototypical Properties" *Artificial Intelligence*, Vol. 33, No. 1, pp. 105-130.

[Doyle79] J. Doyle, "A Truth Maintenance System", *Artificial Intelligence*, Vol. 12, pp 231-273.

[de Kleer86] J. de Kleer, "An Assumption-based TMS", *Artificial Intelligence, Vol. 28, No. 2*, pp. 127-162.

[de Kleer87] J. de Kleer, "Diagnosing Multiple Faults", *Artificial Intelligence, Vol. 32, No. 1*, pp. 97-130.

[Enderton72] H. B. Enderton, *A Mathematical Introduction to Logic*, Academic Press, Orlando.

[Etherington87] D. Etherington, "Formalising Nonmonotonic Reasoning Systems" *Artificial Intelligence, Vol. 31, No. 1*, pp. 41-85.

[Genesereth84] M. R. Genesereth, "The Use of Design Descriptions in Automated Diagnosis", *Artificial Intelligence*, Vol. 24, pp. 411-436.

[Goebel88] R. G. Goebel and S. D. Goodwin, "Applying theory formation to the planning problem" in F. M. Brown (Ed.), *Proceedings of the 1988 Workshop on Nonmonotonic Reasoning*,

[Harman86] G. Harman, *Change in View*, MIT Press.

[Hempel66] C. G. Hempel, *Philosophy of Natural Science*, Prentice Hall, NJ.

[Kautz87] H. A. Kautz, "A Formal Theory of Plan Recognition", Department of Computer Science, University of Rochester, Technical Report TR-215.

[Konolige87] K. Konolige, "On the relationship between default theories and autoepistemic logic", *Proc. IJCAI-87*, pp. 394-401.

[Kowalski79] R. Kowalski, "Algorithm = Logic + Control", *Comm. A.C.M.*, Vol. 22, No. 7, pp. 424-436.

[Lloyd87] J. W. Lloyd, *Foundations of Logic Programming*, Second Edition, Springer-Verlag 1987.

[McCarthy86] J. McCarthy, "Applications of Circumscription to Formalising Common Sense Knowledge", *Artificial Intelligence*, Vol. 28, No. 1, pp. 89-116.

[McDermott87] D. McDermott, "A critique of pure reason", *Computational Intelligence*, Vol. 3, No. 3, pp. 151-160.

[Moore85] R. C. Moore, "Semantical Considerations on Nonmonotonic Logic", *Artificial Intelligence*, Vol. 25, No. 1, pp. 75-94. pp 272-279.

[Neufeld87a] E. M. Neufeld and D. Poole, "Towards solving the multiple extension problem: combining defaults and probabilities", to appear *Workshop on Reasoning with Uncertainty*, Seattle, July 1987.

[Patil81] R. S. Patil, P. Szolovits and W. B. Schwartz, "Causal understanding of patient illness in medical diagnosis", *Proc. IJCAI-81*, pp. 893-899.

[Poole85] D. L. Poole, "On the Comparison of Theories: Preferring the Most Specific Explanation", *Proc. IJCAI-85*, pp.144-147.

[PGA87] D. L. Poole, R. G. Goebel, and R. Aleliunas, "Theorist: a logical reasoning system for defaults and diagnosis", in N. Cercone and G.McCalla (Eds.) *The Knowledge Frontier: Essays in the Representation of Knowledge*, Springer Varlag, New York, 1987, pp. 331-352.

[Poole87] D. L. Poole (Ed.), *Experiments in the Theorist Paradigm: A Collection of student papers on the Theorist Project*, Research Report CS-87-30, Department of Computer Science, University of Waterloo, May.

[Poole88a] D. L. Poole, "A Logical Framework for Default Reasoning", *Artificial Intelligence*, Vol. 36, No. 1, pp. 27-47.

[Poole88b] D. Poole, "Representing knowledge for logic-based diagnosis", *Proceedings of the International Conference of Fifth Generation Computing Systems, 1988*, Tokyo pp. 1282-1290.

[Poole89a] D. Poole, "Normality and faults in logic-based diagnosis", *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, pp. 1304-1310.

[Poole89b] D. L. Poole, "Explanation and Prediction: An Architecture for Default and Abductive Reasoning", to appear *Computational Intelligence*, 1989.

[Popl83] H. E. Popl, Jr., "Heuristic methods for imposing structure on ill structured problems", in P. Szolovits (Ed.), *Artificial Intelligence in Medicine*, AAAS/Westview, Boulder, CO, pp. 119-190.

[Popper62] K. R. Popper, *Conjectures and Refutations: The Growth of Scientific Knowledge*, Basic Books, New York.

[Quine78] W. V. Quine, and J. S. Ullian, *The Web of Belief*, Random House, Yew York, Second Edition.

[Reiter80] R. Reiter, "A Logic for Default Reasoning", *Artificial Intelligence*, Vol. 13, pp 81-132.

[Reiter87] R. Reiter, "A Theory of Diagnosis from First Principles" *Artificial Intelligence*, Vol. 32, No. 1, pp. 57-96.

[Reiter89] R. Reiter and A. K. Mackworth, "A Logical Framework for Depiction and Image Interpretation", to appear,*Artificial Intelligence*

[Weiss78] S. M. Weiss, C. A. Kulikowski, S. Amarel and A. Safir, "A model-based method for computer-aided medical decision making", *Artificial Intelligence 11*, pp. 145-172.