A Logical Framework for Default Reasoning

David Poole
Logic Programming and Artificial Intelligence Group,
Department of Computer Science,
University of Waterloo,
Waterloo, Ontario, Canada, N2L3G1
(519) 888-4443
dlpoole@dragon.waterloo.cdn

October 2, 1987

Abstract

This paper presents a simple logical framework for default reasoning. The semantics is normal first order model theory; instead of changing the logic, the way in which the logic is used is changed. Rather than expecting reasoning to be just deduction (in any logic) from our knowledge, we examine the consequences of viewing reasoning as a very simple case of theory formation. By treating defaults as predefined possible hypotheses we show how this idea subsumes the intuition behind Reiter's default logic. Solutions to multiple extension problems are discussed. A prototype implementation, called *Theorist*, executes all of the examples given.

1 Introduction

There are arguments that we need to have at least the first order predicate calculus in any representation system that is capable of reasoning about individuals and relations amongst individuals [Hayes77, Moore82, Genesereth87]. There has, however, been a perceived problem with using traditional logic

for commonsense reasoning, as it is monotonic (as we add new axioms to a logic the set of logical consequences increases but people seem to revise old opinions with new knowledge). There have been many proposals for augmenting classical logic to allow it to do non-monotonic reasoning [Reiter80, McDermott80, McCarthy86, Moore85].

If one accepts the importance of logic (and the importance of meaning and semantics), there seems to be two approaches to solving this problem of non-monotonicity:

1. The first is to claim that there is obviously something wrong with (say) classical logic and so there is a need to define a new logic to handle nonmonotonic reasoning (e.g., [Reiter80, McDermott80, Moore85, Delgrande87]). If one was to follow this path, then one would define a syntax, semantics and a proof procedure and have theorems of soundness and completeness for this logic.

2. An alternative is to say that there is nothing wrong with classical logic; we should not expect reasoning to be just deduction from our knowledge. Circumscription [McCarthy86] can be seen in this light: defaults are implicit assumptions; we have to make these assumptions explicit by adding the circumscription formula. From the completed knowledge base we can then do deduction.

The proposal in this paper follows this second approach (but in a very different way to circumscription, although some of the details on how to use it are very similar). I argue that, rather than being a problem with logic, nonmonotonicity is a problem of how logic is used. This follows in the spirit of [Israel80].

There are arguments that say that a reasoning system must do some hypothesising and testing. There has been much study of this, particularly in the area of scientific theory formation [Popper59, Quine78, Hempel65, Rescher70].

Here we consider the simplest case of hypothetical reasoning, namely where the user provides the form of possible hypotheses they are prepared to accept in an explanation. We deliberately avoid the most difficult question of science, namely "How can we generate the hypotheses?".

We show that this idea of theory formation from a fixed set of possible hypotheses is a natural and simple characterisation of default reasoning.

Rather than defining a new logic for default reasoning, we would rather say that it is the natural outcome of considering reasoning, not as deduction, but as theory formation. It is logic which tells us what our theory predicts.

The thesis this paper is trying to defend is *If one allows hypothetical reasoning then there is no need to define a new logic to handle nonmonotonic reasoning.*

We give a semantics for our proposal, and show how this can be incorporated into a system for default reasoning. A programming methodology is given which, if followed, allows one to avoid many problematic multiple extensions.

2 Semantics

The intuitive idea is, given a set of observations to be explained, a set of facts known to be true, and a pool of possible hypotheses, to find an explanation which is a set of instances of possible hypotheses used to predict the expected observations (i.e., together with the facts implies the observations) and is consistent with the facts (i.e., does not predict anything known to be false). This *explanation* should be viewed as a "scientific theory" based on a restricted set of possible hypotheses. It is also useful to view the explanation as a scenario in which some goal is true. The user provides what is acceptable in such scenarios.

The framework is presented in terms of the first order predicate calculus, although the idea is independent of the particular logic used. We want a set of possible hypotheses which (together with the facts) implies the goal and does not imply an absurdity (or something known to be false).

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

We assume we are given the following sets

$\mathcal{F}$ is a set of closed formulae, which we are treating as "facts". We assume the facts are consistent. These are intended to be statements which are true in the intended interpretation. More precisely these are statements we are not prepared to give up for some application.

Δ is a set of formulae, called the set of *possible hypotheses*. Any ground instance of these can be used as a hypothesis if it is consistent.

Definition 1 *a* **scenario** *of $\mathcal{F}, \Delta$ is a set $D \cup \mathcal{F}$ where D is a set of ground instances of elements of Δ such that $D \cup \mathcal{F}$ is consistent.*

Definition 2 *If g is a closed formula then an* **explanation** *of g from $\mathcal{F}, \Delta$ is a scenario of $\mathcal{F}, \Delta$ which implies g.*

That is, g is explainable from $\mathcal{F}, \Delta$ if there is a set D of ground instances of elements of Δ such that

$\mathcal{F} \cup D \models g$ and
$\mathcal{F} \cup D$ is consistent

$\mathcal{F} \cup D$ is an explanation of g.

The first condition says that D predicts g, and the second says that D does not predict something which we know is false.

Definition 3 *an* **extension** *of $\mathcal{F}, \Delta$ is the set of logical consequences of a maximal (with respect to set inclusion) scenario of $\mathcal{F}, \Delta$.*

The question that arises is whether this is really a semantics. It is certainly not as complex as other semantics; it is trying to inherit all of its semantics from the first order predicate calculus.

Semantics is the linking of symbols and sentences in our language with the semantic domain. The semantic domain I am interested in is the real world. This is not a subset of the Herbrand Universe, some Kripke structure or some other mathematical structure (though it may have some relation to such structures), but rather a world consisting of trees and chairs and people and diseases. It is this world that my robot must walk in, and this world that my diagnostic program must reason about to determine what is wrong with a patient.

The joy of classical (Tarskian) semantics is that if I choose a denotation for the symbols in my language (i.e., for each symbol in my language I decide that it denotes an individual or relation in the world), and write down in axioms statements true in the world, then all theorems of my axioms will also be true in the world.

Defaults do not exist in the world. "Birds fly" is not a statement which is true or not of a world; in the world there are birds, some fly and some don't. Defaults are assumptions used in building our theory of the world. When we have this theory (a theory being the set of logical consequences of a scenario), then we can talk about semantics and logical consequence. It is these scenarios which have a semantic link to the world.

2.1 Properties of the Definitions

One interesting question is whether scenarios are finite. Note that an extension is not finite nor necessarily finitely axiomatisable (that is, there may not be a finite D such that the extension is the logical consequence of $\mathcal{F} \cup D$).

The following lemma follows directly from the compactness theorem of the first order predicate calculus [Enderton72, p. 136].

Lemma 1 *If g is explainable from $\mathcal{F}, \Delta$ then it is explainable using a finite scenario.*

The following lemma trivially holds.

Lemma 2 (monotonicity) *If $D_1 \subseteq D_2$ and $\mathcal{F} \cup D_1 \models \alpha$ then $\mathcal{F} \cup D_2 \models \alpha$*

There is a very close connection between explainability and being in an extension:

Theorem 3 α is explainable if and only if α is in some extension.

> **Proof:** If α is explainable, there exists a scenario S which implies α. Any extension of S, Δ is an extension of $\mathcal{F}, \Delta$ which contains α by lemma 2. One exists as S is an extension.
>
> Conversely, suppose that α is in some extension. Then $\alpha \in Th(S)$[1] for some scenario S. So, S is consistent and $S \models g$, so S is an explanation for α. □

[1] If A is a set of formulae, $Th(A)$ is the set of logical consequences of A.

3 Default Reasoning

Here we show that, if the possible hypotheses are defaults, the semantics above gives an account for default reasoning.

We are considering defaults to be possible hypotheses which can be used in an explanation. Explainability corresponds to following from the default assumptions. A default is something that someone is prepared to accept as part of an explanation of why something is expected to be true.

When considering possible hypotheses to be defaults, the thing to be explained is intended to be something which we are predicting, rather than something we are actually observing.

Example 1 Consider the statement "birds fly". This can be expressed as the default:

$$\Delta = \{bird(x) \Rightarrow flies(x)\}$$

This means that, for a particular value b of x, if $bird(b)$ can be proven, so can $flies(b)$, as long as the default has not been contradicted for that particular b. If $bird(b)$ and $\neg flies(b)$ are provable, the default is contradicted for that particular b, and so cannot be used.

Suppose that, together with the above assumption, we have the facts that emus are birds which do not fly, Polly is an emu and Tweety is a bird. This can be expressed as:

$$\begin{aligned}\mathcal{F} = \{\ & \forall x\ emu(x) \Rightarrow bird(x),\\ & \forall x\ emu(x) \Rightarrow \neg flies(x),\\ & emu(polly),\\ & bird(tweety)\}\end{aligned}$$

$flies(tweety)$ is explainable by

$$\{bird(tweety) \Rightarrow flies(tweety)\}$$

which is consistent with the facts. This should be translated as meaning that we expect tweety to fly as tweety is a bird and we know birds, by default fly, and there is no reason to believe that tweety doesn't fly. $flies(polly)$ is potentially explainable by

$$\{bird(polly) \Rightarrow flies(polly)\}$$

but this is not consistent with the facts, as its negation can be proved. So $flies(polly)$ is not explainable.

4 A comparison with Reiter's logic

We make two comparisons with Reiter's default logic [Reiter80]. In this section we show that the above logic is a restricted case of his normal defaults, and section 8 shows how to get the power of more general defaults without sacrificing the simple semantics.

The default $w \in \Delta$ is a syntactic variant of Reiter's normal default

$$\frac{: M\ w}{w}$$

Reiter's logic for closed defaults is defined in terms of "extensions". In this section we show that our own notion of an extension (definition 3) in terms of a maximal scenario corresponds to Reiter's definition in terms of fixed points.

The following theorem parallels the definition of a closed extension given in [Reiter80]. This shows that our definition of an extension in terms of maximal theories corresponds to his definition in terms of fixed points.

Theorem 4 Suppose E is an extension of $\mathcal{F}, \Delta$. Then

1. $\mathcal{F} \subseteq E$
2. $Th(E) = E$
3. If α is a ground instance of an element of Δ, and $\neg\alpha \notin E$ then $\alpha \in E$

Furthermore E is minimal with respect to the above three properties.

> **Proof:** If E is an extension of of $\mathcal{F}, \Delta$, then $E = Th(\mathcal{F} \cup D)$ for some (possibly infinite) maximal set D of ground instances of elements of Δ.
>
> 1. follows since $\mathcal{F} \subseteq Th(\mathcal{F} \cup D)$ for any D
> 2. follows from the properties of Th, since $Th(Th(S)) = Th(S)$ for any S.

3. Suppose α is a ground instance of a possible hypothesis, and $\neg\alpha \notin E$. If $\alpha \notin E$, then $\mathcal{F} \cup D \cup \{\alpha\}$ is a strict superset of $\mathcal{F} \cup D$ and is consistent (as $\neg\alpha \notin Th(E)$) which contradicts the maximality of E.

To show $E = Th(\mathcal{F} \cup D)$ is minimal with respect to the above three properties, suppose $E' \subseteq E$, $E \not\subseteq E'$; and E' is a set with the above three properties. There must be some ground instance α of an element of Δ such that $\alpha \in D$ and $\alpha \notin E'$ (as if every element of D is in E', we have $\mathcal{F} \cup D \subseteq E'$ and so $E = Th(\mathcal{F} \cup D) \subseteq Th(E') = E'$). α is a ground instance of a possible hypothesis, and $\neg\alpha \notin E'$ (as $E' \subseteq E$ and $\neg\alpha \notin E$ (as $\alpha \in E$ and E is consistent)). But $\alpha \notin E'$, contradicting 3, so no such E' can exist. □

5 Naming Defaults

One of the things that turns out to be useful is the ability to name defaults. These will be names parameterised by the free variables in a default. There are three reasons for wanting to do this

1. They make it better for printing explanations;

2. When implementing our system we need only worry about the name, and can essentially compile out the structure of defaults. I show (theorem 5) that no power is lost by restricting the system to very simple forms of defaults.

3. We want to be able to explicitly talk about a default so that we can, for example, explicitly say when it is not applicable. This is useful for solving multiple extension problems.

If $w[\overline{x}]$ is a default with free variables $\overline{x} = x_1, ..., x_n$, then we can name w with $p_w(x_1, ..., x_n)$ where p_w is an n-ary predicate symbol not appearing elsewhere in the system (i.e., in $\mathcal{F}$ and Δ).

In this section, we want to show how to use a name. Essentially we make Δ contain only the name of defaults, and have the name implying the default as a fact. First, we want to show that restricting Δ to names does not reduce

the power, or change the semantics of the system. We then discuss how to use the name to solve multiple extension problems. The use of naming is very closely related to McCarthy's abnormality predicate [McCarthy86], and indeed the translation is similar to that of [Grosof84] and the use of the $APPL$ predicate of [Brewka86].

Given $\mathcal{F}$ and Δ, define

$$\Delta' = \{p_w(\overline{x}) \text{ s.th. } w[\overline{x}] \in \Delta\}$$

$$\mathcal{F}' = \mathcal{F} \cup \{\forall \overline{x}\ p_w(\overline{x}) \Rightarrow w[\overline{x}] \text{ s.th. } w[\overline{x}] \in \Delta\}$$

That is $\mathcal{F}'$, Δ' is the system where all defaults are named, and they replace the defaults in Δ. The following theorem shows that this can always be done without any unexpected side effects.

Theorem 5 (Renaming) If the predicate symbols p_w do not appear in $\mathcal{F}$, Δ or g, then g is explainable from $\mathcal{F}$ and Δ if and only if g is explainable from $\mathcal{F}'$ and Δ'.

Proof: Suppose g is explainable from $\mathcal{F}$ and Δ, then there is some explanation $D = \{w[\overline{c}]\}$ where $w[\overline{x}] \in \Delta$, such that $\mathcal{F} \cup D \models g$ and $\mathcal{F} \cup D$ is consistent.

Let $D' = \{p_w(\overline{c}) \text{ s.th. } w[\overline{c}] \in D\}$. To show $\mathcal{F}' \cup D' \models g$. Suppose, on the contrary, $\mathcal{F}' \cup D' \cup \{\neg g\}$ is true in interpretation I. Now $\mathcal{F}$ and $\neg g$ are true in I. For each $w[\overline{c}]$ in D, we know $p_w(\overline{c})$ and $p_w(\overline{x}) \Rightarrow w[\overline{x}]$ is true in I and so $w[\overline{c}]$ is true in I, which makes I a model of $\mathcal{F} \cup D \cup \{\neg g\}$, a contradiction to $\mathcal{F} \cup D \models g$. So no such model can exist, so $\mathcal{F}' \cup D' \models g$.

To show $\mathcal{F}' \cup D'$ is consistent, suppose I is a model of $\mathcal{F} \cup D$. Let I' be an interpretation which is identical to I except that $p_w(\overline{c})$ is true in I' exactly when $w(\overline{c})$ is true in I. Then I' is a model for $\mathcal{F}' \cup D'$, as $\mathcal{F}$ is true in I' (as $p_w(\overline{c})$ does not appear in $\mathcal{F}$, and $\mathcal{F}$ is true in I), $p_w(\overline{x}) \Rightarrow w[\overline{x}]$ is true in I' by definition, and D' is true in I' as D is true in I.

For the only-if half of the theorem, suppose g is explainable from $\mathcal{F}'$ and Δ'. Then there is some D' such that $\mathcal{F}' \cup D' \models g$ and $\mathcal{F}' \cup D'$ is consistent. Let $D = \{w[\overline{c}] \text{ s.th. } p_w(\overline{c}) \in D'\}$. By a dual argument to that above, $\mathcal{F} \cup D \models g$ and $\mathcal{F} \cup D$ is consistent. □

This shows that we can work in the named system as well as in the general system, and that we are not restricting the power of the system by only having atoms as defaults.

6 A Programming Language for Defaults

So far we have given a semantics for default reasoning. In this section we present a programming language for default reasoning (called *Theorist*), and give a programming methodology, such that problems of undesirable multiple extensions can be solved.

We define a very simple language, with the following declarations[2]:

fact w.
: where w is a formula, means "$\forall w$"$\in \mathcal{F}$.

default n.
: where n is a name (predicate with only free variables as arguments) means $n \in \Delta$.

default $n : w$.
: where w is a formula, and n is a name with the same free variables as w, means that w is a default, with name n. Formally this means that $n \in \Delta$ and "$\forall n \Rightarrow w$"$\in \mathcal{F}$.

explain g.
: where g is a formula, asks whether we can explain $\exists g$ from the $\mathcal{F}$ and Δ. A consistent explanation is returned.

Within the programming language, we follow the Prolog convention of having variables in uppercase, constants in lower case, and unbound variables being implicitly universally quantified. We also use the symbol "$\leftarrow$" as material implication, read "if".

Example 2 the birds flying example above could be given as:

[2] $\forall w$ is the universal closure of w. That is, if w is a formula with free variables $\overline{v}$, then $\forall w$ means $\forall \overline{v}\ w$. Similarly $\exists w$ is the existential closure of w, that is $\exists \overline{v}\ w$.

default $birdsfly(X) : flies(X) \leftarrow bird(X)$.
fact $\neg flies(X) \leftarrow emu(X)$.
fact $bird(X) \leftarrow emu(X)$.
fact $bird(tweety)$.
fact $emu(polly)$.

Here the query

explain $flies(tweety)$.

comes back with the answer "yes", assuming $\{birdsfly(tweety)\}$. Note that the explanation gives the assumptions used to support our prediction. It tells us the reason we are predicting the result. We predict Tweety flies, as Tweety is a bird, and birds fly.

The query

explain $flies(polly)$.

comes back with the answer "no" (meaning that it could not be explained).

6.1 Programming Methodology and Multiple Extension Problems

The idea of writing programs in this system is to add generalised knowledge as defaults, and to add knowledge that we know is true as facts. Also we add knowledge as to when a default is not applicable (by using the name of a default), as follows

If we have formula w which we want to use as a default, and we know that it is not applicable under condition c, we can give the system,

default $n : w$.
fact $\neg n \leftarrow c$.

We are saying that we can assume n and so infer w. This is inconsistent (and so cannot be used) if we can prove $\neg w$ or if we prove c. The second rule corresponds to the *cancellation of inheritance* axioms of [McCarthy86, p. 93].

Example 3 Consider the following:

default *mammals-don't-fly*$(X) : \neg$*flies*$(X) \leftarrow$ *mammal*(X).
default *bats-fly*$(X) :$ *flies*$(X) \leftarrow$ *bat*(X).
default *dead-things-don't-fly*$(X) : \neg$*flies*$(X) \leftarrow$ *dead*(X).
fact *mammal*$(X) \leftarrow$ *bat*(X).
fact *bat*$(dracula)$.
fact *dead*$(dracula)$.

We can explain $\neg flies(dracula)$ with

$$\{\textit{mammals-don't-fly}(dracula)\}$$

That is, we predict that Dracula does not fly because Dracula is a mammal, and mammals, by default, don't fly.

We can explain $flies(dracula)$ with

$$\{\textit{bats-fly}(dracula)\}$$

That is, we expect Dracula flies, because he is a bat, and bats typically can fly.

We can also explain $\neg flies(dracula)$ with

$$\{\textit{dead-things-don't-fly}(dracula)\}$$

We can predict that Dracula doesn't fly as he is a dead bat, and dead bats don't fly by default.

If we don't like the first explanation because we don't want the first default to be applicable to bats, we can add

fact $\neg$*mammals-don't-fly*$(X) \leftarrow$ *bat*(X).

Thus if we can prove X is a bat then we cannot use the default that X doesn't fly because X is a mammal and mammals don't fly.

If we don't want the second explanation for dead bats we can add

fact $\neg$*bats-fly*$(X) \leftarrow$ *dead*(X).

In the resulting system, we can not explain $flies(dracula)$ and can only explain $\neg flies(dracula)$, for the right reason: because he is a dead bat, and dead bats typically don't fly.

Example 4 This example is from [Reiter81] where it is claimed that normal defaults are not adequate to allow explicit exceptions to rules

default $\mathit{unemp\text{-}if\text{-}st}(X) : \neg \mathit{employed}(X) \leftarrow \mathit{uni\text{-}student}(X)$.
default $\mathit{emp\text{-}if\text{-}ad}(X) : \mathit{employed}(X) \leftarrow \mathit{adult}(X)$.
default $\mathit{ad\text{-}if\text{-}st}(X) : \mathit{adult}(X) \leftarrow \mathit{uni\text{-}student}(X)$.
fact $\mathit{uni\text{-}student}(\mathit{paul})$

we can explain $\mathit{employed}(\mathit{paul})$ with

$$\{\mathit{emp\text{-}if\text{-}ad}(\mathit{paul}), \mathit{ad\text{-}if\text{-}st}(\mathit{paul})\}$$

That is, we can predict that paul is employed as he is a student, and so, by default an adult and so by default is employed.

We can explain $\neg \mathit{employed}(\mathit{paul})$ with

$$\{\mathit{unemp\text{-}if\text{-}st}(\mathit{paul})\}$$

If we don't like the first explanation because we don't want the *emp-if-ad* default to be applicable to university students, we can add

fact $\neg \mathit{emp\text{-}if\text{-}ad}(X) \leftarrow \mathit{uni\text{-}student}(X)$.

This makes the first explanation inconsistent.

Note that here we are talking about predicting using defaults rather than explaining observations. In this case it is reasonable that there are some cases where we can explain some proposition and its negation. This means that we have evidence for both predictions and so, on semantic grounds none is preferred (There may, however be heuristic grounds, e.g. probability, to prefer one explanation over the other [Neufeld87]).

Example 5 Consider, for example, the quaker-republican example of [Reiter81]

default $\mathit{quakers\text{-}are\text{-}pacifists}(X) : \mathit{pacifist}(X) \leftarrow \mathit{quaker}(X)$.
default $\mathit{republican\text{-}so\text{-}not\text{-}pacifist}(X) : \neg \mathit{pacifist}(X) \leftarrow \mathit{republican}(X)$.
fact $\neg \mathit{quakers\text{-}are\text{-}pacifists}(X) \leftarrow \mathit{republican}(X)$.
fact $\neg \mathit{republican\text{-}so\text{-}not\text{-}pacifist}(X) \leftarrow \mathit{quaker}(X)$.
fact *republican(ron).*
fact *quaker(george).*
fact *quaker(dick).*
fact *republican(dick).*

Here we can explain $pacifist(george)$ with the explanation $\{$*quakers-are-pacifists*$(george)\}$. We can explain $\neg pacifist(ron)$ with the explanation $\{$*republican-so-not-pacifist*$(ron)\}$. We cannot explain anything about Dick's pacifism.

7 Pruning the set of scenarios

The preceding section showed how many cases where it was deemed necessary to have semi-normal defaults [Reiter81], can be solved without adding anything beyond a simple form of normal defaults. There is however, a problem which still arises.

Example 6 ([Etherington87b]) Consider the problem of being able to assume that some person is a suspect. If we can show they are not guilty, then they should not be a suspect, however we should not conclude that they are guilty just because they are a suspect. In the first case we want some constraint on the possible scenarios, namely $\neg guilty(X) \Rightarrow \neg suspect(X)$. We want to eliminate any scenario in which someone is not guilty and still a suspect. However we do not want this constraint as a fact, otherwise we can conclude that someone is guilty because they are a suspect.

It seems as though we need a way to prune the set of scenarios which is not just adding more facts.

A similar problem arises when cancelling defaults:

Example 7 Assume we have the following Theorist fragment

default $birdsfly(X) : flies(X) \leftarrow bird(X)$.
fact $\neg birdsfly(X) \leftarrow emu(X)$.

This is meant to say that we can use $birdsfly(X)$ to predict some bird flying, but that this is not applicable if we can prove the X is an emu. There are, however, other predictions which can be drawn from these.

From this we may assume $birdsfly(b)$ for any b. This predicts $\neg emu(b)$. Thus we can explain $\neg emu(b)$ for any b. One can argue that this is reasonable, since if we can assume that birds fly and that this doesn't apply to emus, we are implicitly assuming that anything is not an emu.

Using the default we can also explain $\neg bird(b)$ from $\neg flies(b)$. This also could be argued to be reasonable, in much the same way as contrapositives are sensible when doing deduction.

Example 8 Suppose that we want to have the default that if X is a person then we can assume that they are not a diabetic. This can be expressed as

$$\textbf{default } \textit{not-diabetic-if-person}(X) : \neg \textit{diabetic}(X) \leftarrow person(X).$$

This can be used to explain $\neg diabetic(robin)$ from $person(robin)$ but can also be used to explain $\neg person(john)$ from $diabetic(john)$.

There seems to be two points of view of what is happening here. The first is saying that there is nothing wrong here. When you are assuming that if something was a bird it would fly, you are implicitly assuming that it is not an emu. For the first example there is an incorrect interpretation of the term "guilty"; "if someone is not guilty they are not a suspect" is false in the intended interpretation. If we instead mean "possibly guilty" then there is no problem. The problem with example 8 is that the restriction of $person(X)$ on the right hand side is irrelevant to the default (if it was not irrelevant then the conclusion that non diabetics are not human is reasonable). In fact all of the examples of [Reiter81] and [Etherington83] work using just the naming conventions described in the last section.

There is also the point of view which says that the above derivations are wrong. Users of the Theorist system have found that they needed a way to say "this default should not be applicable in this case", without any side effect of doing this. In this section we define the notion of constraints which allow one to do exactly this. They have been found to be a very useful mechanism in practice. Other systems overcome such derivations by allowing the defaults as rules which can only be used in one direction and have explicit exceptions [Reiter80] or by having fixed and variable predicates [McCarthy86].

The idea is to define a set of constraints used to prune the set of scenarios. They are just used to reject scenarios and cannot be used to explain anything. Constraints are formulae with which scenarios must be consistent.

We introduce a set $\mathcal{C}$ of closed formulae called the set of **constraints**[3]. The definition of scenario is revised as follows:

[3] [Gagné87] also describes such conditions, and calls these "the restrictive facts".

Definition 4 *a* **scenario** *of $\mathcal{F}, C, \Delta$ is a set $D \cup \mathcal{F}$ where D is a set of ground instances of elements of Δ such that $D \cup \mathcal{F} \cup C$ is consistent.*

The definitions of explainable and extension are correspondingly changed.

The following is a corollary of theorem 3. The proof is a paraphrase of the proof for theorem 3.

Corollary 6 *In the system with constraints α is explainable if and only if α is in some extension.*

We extend the programming language to have the declaration:

constraint w.

where w is some formula (free variables are implicitly universally quantified) to mean "$\forall w$"$\in \mathcal{C}$.

We can use constraints to prevent a default d being applicable under circumstances c, without allowing an explanation of $\neg c$ by giving

constraint $\neg d \leftarrow c$.

All this constraint does is to reject any scenario which implies both d and c. It has no other affect.

We can prevent the use of the contrapositive of

default $d : c \leftarrow b$.

(that is, we prevent the use of default d to derive $\neg b$ from $\neg c$) by adding the constraint

constraint $\neg d \leftarrow \neg c$.

If we know that c is not true, we cannot use default d. The only affect this constraint has is to not allow any scenario which implies both $\neg c$ and d.

Note that here $\mathcal{F} \cup D$ still has first-order semantics,so that $a \leftarrow b$ and $\neg b \leftarrow \neg a$ are logically equivalent. We have added constraints to restrict the circumstances in which the default can be used.

Example 9 From

default $birdsfly(X) : flies(X) \leftarrow bird(X)$.
constraint $\neg birdsfly(X) \leftarrow \neg flies(X)$.
fact $bird(polly)$.
fact $\neg flies(bruce)$.

we can explain $flies(polly)$, but cannot explain $\neg bird(bruce)$ (as we could if the constraint was not there) or explain $flies(david)$ (as we could if the constraint was a fact).

Example 10 Consider example 3. We can change it to not allow contrapositives or conclusions that something is not a bat or is not dead, by specifying the following:

default *mammals-don't-fly*$(X) : \neg$*flies*$(X) \leftarrow$ *mammal*(X).
constraint $\neg$*mammals-don't-fly*$(X) \leftarrow flies(X)$.
default *bats-fly*(X) : *flies*$(X) \leftarrow$ *bat*(X).
constraint $\neg$*bats-fly*$(X) \leftarrow \neg$*flies*(X).
constraint $\neg$*mammals-don't-fly*$(X) \leftarrow$ *bat*(X).
default *dead-things-don't-fly*$(X) : \neg$*flies*$(X) \leftarrow$ *dead*(X).
constraint $\neg$*dead-things-don't-fly*$(X) \leftarrow flies(X)$.
constraint $\neg$*bats-fly*$(X) \leftarrow$ *dead*(X).
fact *mammal*$(X) \leftarrow$ *bat*(X).
fact $mammal(bruce)$.
fact $bat(paul)$.
fact $bat(dracula)$.
fact $dead(dracula)$.

Here we can conclude that Bruce cannot fly, that Paul can fly and that Dracula cannot fly. For each of these we cannot also explain their negations. We also claim that there are not any unexpected side effects.

8 Reiter's General Defaults

In section 4 we showed how our defaults can be seen as a restriction of Reiter's normal defaults. In this section we want to argue that the extra expressiveness of Reiter's general defaults are not needed.

Reiter's general defaults are of the form:

$$\frac{\alpha(\overline{x}) : M\ \beta_1(\overline{x}), ..., M\ \beta_n(\overline{x})}{\gamma(\overline{x})}$$

(where $\overline{x}$ is the set of free variables). This is intended to mean that if $\alpha(\overline{c})$ is proven and the $\beta_i(\overline{c})$ are consistent then $\gamma(\overline{c})$ can be inferred.

This can be simulated in Theorist by creating the name M_{β_i} for each β_i and the relations $M_{\beta_i}(\overline{x})$ which we can assume any instance of as long as we cannot prove the corresponding instance of $\neg\beta_i(\overline{x})$. This is done by having for each i,

default $M_{\beta_i}(\overline{x})$.
constraint $\neg M_{\beta_i}(\overline{x}) \leftarrow \neg\beta_i(\overline{x})$.

and the fact

fact $\gamma(\overline{x}) \leftarrow M_{\beta_1}(\overline{x}) \wedge ...M_{\beta_n}(\overline{x}) \wedge \alpha(\overline{x})$.

This is very close to [Reiter80] if the defaults are semi-normal. For example, the default

$$\frac{\alpha(\overline{x}) : M\ \beta(\overline{x}) \wedge \gamma(\overline{x})}{\gamma(\overline{x})}$$

can be approximated using the above technique by creating the name $M_{\beta\gamma}$ with the following definitions

default $M_{\beta\gamma}(\overline{x}) : \gamma(\overline{x}) \leftarrow \alpha(\overline{x})$.
constraint $\neg M_{\beta\gamma}(\overline{x}) \leftarrow \neg\beta(\overline{x})$.
constraint $\neg M_{\beta\gamma}(\overline{x}) \leftarrow \neg\gamma(\overline{x})$.

The first constraint says that we cannot use any instance of the default for which we can prove $\neg\beta$. The second prevents the use of the contrapositive of the default.

This translation is not exact. The following is an example where they produce different answers. I would argue that Reiter's defaults gives unintuitive results when they differ.

Example 11 Consider the case where we have, by default, emus and ostriches both run, and we know that Polly is either an emu or an ostrich (they do look similar). In Reiter's notation this can be written as

$$emu(polly) \vee ostrich(polly)$$

$$\frac{emu(X) : M\ runs(X)}{runs(X)}$$

$$\frac{ostrich(X) : M\ runs(X)}{runs(X)}$$

we cannot derive $runs(polly)$ in Reiter's system as we cannot prove the antecedent of either default. If we consider the Theorist translation we have

fact $emu(polly) \vee ostrich(polly)$.
default $emus\text{-}run(X) : runs(X) \leftarrow emu(X)$.
constraint $\neg emus\text{-}run(X) \leftarrow \neg runs(X)$.
default $ostriches\text{-}run(X) : runs(X) \leftarrow ostrich(X)$.
constraint $\neg ostriches\text{-}run(X) \leftarrow \neg runs(X)$.

From this we can explain $runs(polly)$ with

$$\{emus\text{-}run(polly), ostriches\text{-}run(polly)\}$$

which is consistent (even if we have emus and ostriches are disjoint classes).

Note that the use of constraints is not what is important in this example, but rather the special status of the preconditions of Reiter's defaults.

Note also that slight changes to the representation of the domain makes Reiter's system give different answers. For example, if there was a class of *big birds with short feathers*, which covered both emus and ostriches, and they, by default run, then Reiter's system can explain that Polly runs.

The second difference is that we do not have the problems of semi-normal defaults not having extensions [Etherington87a]. Etherington give the following example:

$$\frac{: A \wedge \neg B}{A}, \frac{: B \wedge \neg C}{B}, \frac{: C \wedge \neg A}{C}$$

Using the above translation from general defaults, this is translated into[4]

default $manb : a$.
default $mbnc : b$.

[4]Note that we don't need the constraint that $\neg manb \leftarrow \neg a$ as we have $a \leftarrow manb$ as a fact.

default $mcna : c$.
constraint $\neg manb \leftarrow b$.
constraint $\neg mbnc \leftarrow c$.
constraint $\neg mcna \leftarrow a$.

Here we have three extensions, one for each case of assuming one of the defaults. For example, we can assume $manb$, which allows us to predict a. The first constraint says that we have implicitly assumed $\neg b$. The problem with the semi-normal defaults not having an extension arises because they have not allowed the recording of what assumptions have been implicitly made.

Theorem 7 (Monotonicity of Defaults) *Adding defaults can only increase the number of things explainable, adding constraints can only decrease the number of things explainable.*

> **Proof:** Suppose g is explainable from $\mathcal{F}$, Δ, $\mathcal{C}$. Then there is some D, a set of instances of elements of Δ such that $\mathcal{F} \cup D \models g$ and $\mathcal{F} \cup D \cup \mathcal{C}$ is consistent. If $\Delta \subseteq \Delta'$ then g is explainable from $\mathcal{F}$, Δ', $\mathcal{C}$ using the same D. If $\mathcal{C}' \subseteq \mathcal{C}$, then g is explainable from $\mathcal{F}$, Δ, $\mathcal{C}'$ as $\mathcal{F} \cup D \models g$ and $\mathcal{F} \cup D \cup \mathcal{C}'$ is consistent. $\square$

Corollary 8 *there is always an extension if $\mathcal{F} \cup \mathcal{C}$ is consistent.*

One useful case for which we can guarantee this is if

1. the explicit facts (those added with the *fact* declaration) are consistent,
2. all of the constraints are of the form $\neg d \leftarrow c$ where d is a default name, and
3. there are no default names appearing in the explicit facts. This means the only default names in $\mathcal{F}$ are there by virtue of being in a *default* declaration.

This is because the interpretation which is the model of the explicit facts, but with all default names $false$ is a model of $\mathcal{F} \cup \mathcal{C}$.

The translation of Reiter's defaults follows this convention, and so if the facts are consistent, there is always an extension.

Theorem 7 has consequences beyond just ensuring that we have extensions. Consider the following example:

Example 12 Suppose we have a system with the defaults

$$\frac{A : B}{B}, \frac{A : C}{C}, \frac{C : \neg B \wedge D}{D}$$

According to the semantics of [Reiter80] there is one extension containing B and C and not D. The first default forces the third default to be inapplicable.

In our translation, we have two extensions, one containing B and C, and the other containing C and D. The second extension is obtained by using the third default which implicitly assumed that B is false, and so making the first default inapplicable.

8.1 Inheritance Networks

Another interesting feature of our translation of semi-normal defaults is in the translation of inheritance networks with exceptions [Etherington87a]. We can follow Etherington's translations, but do the translation defined above from semi-normal defaults into Theorist[5]. This is done by having the defaults named, so that the default IS-A from A to B becomes the declaration

default $b\text{-}if\text{-}a(X) : b(X) \leftarrow a(X).$
constraint $\neg b\text{-}if\text{-}a(X) \leftarrow \neg b(X).$

the default ISN'T-A similarly has the translation

default $not\text{-}b\text{-}if\text{-}a(X) : \neg b(X) \leftarrow a(X).$
constraint $\neg not\text{-}b\text{-}if\text{-}a(X) \leftarrow b(X).$

The exception links translate into into constraints. The arc from node c to the arc representing the default named d (given the name in the default above) becomes

constraint $\neg d \leftarrow c.$

Here we have modular statements with each arc in the inheritance hierarchy becoming a small number of rules in the default system. There are no translated rules which depend on other arcs. This translation thus overcomes the objections that Touretzky has to Etherington's translation [Etherington87a, p. 63-64].

[5]We assume that the reader is familiar with the translation in [Etherington87a, p. 56]. We use only the different convention of having variables in upper case.

9 Implementation

The implementation of Theorist is very simple, is closely related to the top down default proof of [Reiter80], and is described elsewhere [Poole87a].

To find an explanation of g, we attempt to prove g with the facts, $\mathcal{F}$, and possible hypotheses, Δ, as axioms. We make D_0 the set of instances of elements of Δ used in the proof, and make D_1 a grounding of D_0 (replacing all free variables with unique constants). We then know

$$\mathcal{F} \cup D_1 \models g$$

We can use a complete theorem prover to prove that $\mathcal{F} \cup \mathcal{C} \cup D_1$ is consistent, by failing to prove it is inconsistent. In general this is undecidable, but has not been a problem in the domains we have considered.

There is currently an interpreter for Theorist [Poole87a], based on a complete theorem prover written in Prolog. There is also a compiler which transforms Theorist facts, constraints and defaults into Prolog. We are currently working on a system based on truth maintenance and dependency directed backtracking [Doyle79, de Kleer85]. We are also building a system which does concurrent consistency checking.

We have avoided the problems that arise with existential and universally quantified variables in defaults [Poole87b] by requiring that we only use ground instances of defaults in theories.

10 Conclusion and Future Work

We have presented here a thesis that the problem of default reasoning is not a problem with logic, but with how logic is used. Not many people would disagree with the proposition that we need to do some sort of hypothetical reasoning; we are investigating what can be done with a very simple form, namely where we have the user providing the possible hypotheses they are prepared to accept in an explanation. We have shown how the use of theory formation, using constrained forms of hypotheses, can be used as a basis for default reasoning.

A simple "semantics" was proposed and compared in detail to the proposal of [Reiter80]. This has the advantage of simplicity, of being defined in

terms of a semantic characterisation rather than just a proof procedure, and of not needing to change the underlying semantics to do default reasoning.

This approach gives us a neat way to examining other problems. One of the interesting questions concerns the comparison of scenarios. That is, determining when one scenario is "better" than another for some purpose. [Poole85] argues that the problem of inheritance in semantic nets can be best done by preferring the most specific theory. That is, when there is specific knowledge and more general knowledge available, then we prefer to use the most specific knowledge. [Goebel87] shows how the multiple extension problem in axiomatising the frame axioms can be solved in the Theorist framework by preferring the chronologically maximally persistent theory. For a diagnostic system we may prefer the most likely explanation [Neufeld87].

This theory provides the basis for the Theorist system [Poole87a], which we are using for a variety of domains, including diagnosis, building user models, problem solving by analogical reasoning, pronoun resolution and diagnosis of students with learning disabilities.

Acknowledgements

This work was started while the author was at the Australian National University. Special thanks to my supervisor, Robin Stanton of the Australian National University. Thanks also to Hugh MacKenzie and Peter Creasy for many discussions and Randy Goebel, Eric Neufeld, Paul Van Arragon, Romas Aleliunas, Bruce Kirby, Scott Goodwin and Denis Gagné for valuable arguments, discussions and feedback. This research was supported under NSERC grant A6260.

References

[Brewka86] G. Brewka, "Tweety – Still Flying: Some Remarks on Abnormal Birds, Applicable Rules and a Default Prover", *Proc. AAAI-86*, pp. 8-12.

[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 Kleer85] J. de Kleer, "An Assumption-based TMS", *Artificial Intelligence, Vol. 28, No. 2*, pp. 127-162.

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

[Etherington83] D. Etherington and R. Reiter, "On Inheritance Hierarchies With Exceptions", *Proc. AAAI-83*, pp. 104-108.

[Etherington87a] D. Etherington, "Formalising Nonmonotonic Reasoning Systems" *Artificial Intelligence, Vol. 31, No. 1*, pp. 41-85.

[Etherington87b] D. Etherington, *Reasoning from incomplete information*, Research notes in AI, Pitman, London, 1987.

[Gagné87] D. Gagné, *The Multiple Extension Problem Revisited*, in Technical Report, CS-87-30, Department of Computer Science, University of Waterloo.

[Genesereth87] M. Genesereth and N. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan-Kaufmann, Los Altos, California.

[Goebel87] R. G. Goebel and S. D. Goodwin, "Applying theory formation to the planning problem" in F. M. Brown (Ed.), *Proceedings of the 1987 Workshop on The Frame Problem in Artificial Intelligence*, Morgan Kaufmann, pp. 207-232.

[Grosof84] B. Grosof, "Default Reasoning as Circumscription", *Proc. AAAI Non-monotonic Reasoning Workshop*, pp. 115-124.

[Hayes77] P. J. Hayes, "In Defence of Logic", *Proc. IJCAI-77*, pp. 559-565.

[Hempel65] C. G. Hempel, *Aspects of Scientific Explanations and other essays in the Philosophy of Science*, The Free Press, New York.

[Israel80] D. J. Israel, "What's wrong with Non-monotonic Logic", *AAAI-80*, pp. 99-101.

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

[McDermott80] D. V. McDermott, and J. Doyle, "Non Monotonic Logic 1", *Artificial Intelligence*, Vol. 13, pp 41-72.

[Moore82] R. C. Moore, "The Role of Logic in Knowledge Representation and Commonsense Reasoning", *Proc. AAAI-82*, pp. 428-433.

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

[Neufeld87] 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.

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

[Poole87a] 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; also Research Report CS-86-06, Department of Computer Science, University of Waterloo, 16 pages, February 1986.

[Poole87b] D. L. Poole, "Variables in Hypotheses", to appear, *IJCAI-87*.

[Popper59] K. R. Popper, *The Logic of Scientific Discovery*, 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.

[Reiter81] R. Reiter, and G. Criscuolo, "On Interacting Defaults", *Proc. IJCAI-81*, pp.270-276.

[Rescher70] N. Rescher, *Scientific Explanation*, Collier-MacMillan Canada, Toronto.

[Touretzky84] D. S. Touretzky, "Implicit Ordering of Defaults in Inheritance Systems", *Proc AAAI-84*, pp 322-325.