Prediction is Deduction but Explanation is Abduction

Murray Shanahan
Imperial College
Department of Computing,
180 Queen's Gate,
London SW7 2BZ.
England.

Abstract

This paper presents an approach to temporal reasoning in which prediction is deduction but explanation is abduction. It is argued that all causal laws should be expressed in the natural form *effect if cause.* Any given set of laws expressed in this way can be used for both forwards projection (prediction) and backwards projection (explanation), but *abduction* must be used for explanation whilst deduction is used for prediction. The approach described uses a shortened form of Kowalski and Sergot's Event Calculus and incorporates the assumption that properties known to hold must have explanations in terms of events. Using abduction to implement this assumption results in a form of default persistence which correctly handles problems which have troubled other formulations. A straightforward extension to SLD resolution is described which implements the abductive approach to explanation, and which complements the well-understood deductive methods for prediction.

1. Introduction

Temporal reasoning involves both *prediction* and *explanation.* Prediction is projection forwards from causes to effects whilst explanation is projection backwards from effects to causes. That is, prediction is reasoning from events to the properties and events they cause, whilst explanation is reasoning from properties and events to events that may have caused them. Although it is clear that a complete framework for temporal reasoning should provide facilities for solving both prediction and explanation problems, prediction has received far more attention in the temporal reasoning literature than explanation.

Frequently, outside of the temporal reasoning literature, explanation problems are seen as *deductive.* Domain knowledge is captured in a theory T, the effects that require explanation are represented by a set of sentences *A,* and the causes of *A* are amongst the logical consequences *G* such that *T U G.* For example, in Mycin a set of rules *T* relates symptoms to diseases. Each rule is roughly of the form *cause if effect.* The symptoms are represented by *A* and the disease which causes those symptoms is a logical consequence of *T U* A

Mycin rules look rather peculiar, since they invert the relationship between cause and effect. This is because Mycin treats explanation as deduction rather than abduction. This kind of "compilation" of causal laws into inverted implications is counter-intuitive and is not always appropriate or possible. Furthermore, a set of Mycin rules is no good for predicting what symptoms are caused by a given disease, even though intuitively it is clear that if the rules ir *T* adequately capture the domain, they should be equally good for both prediction and explanation.

This confusion of explanation with deduction is possible not only with Mycin's shallow sort of causal reasoning, but also with temporal reasoning in general, in which time is represented explicitly. Domain knowledge is captured in a theory *T,* events and properties are represented by a set of sentences *Δ,* and amongst the logical consequences *G* such that *T* U *Δ* = *G* are both predictions and explanations. That is, *G* represents projections both forwards and backwards from *A.*

An alternative and more natural approach is one in which prediction is deductive but explanation is strictly *abductive.* Causal laws are captured in a theory T, and each law has the more intuitive form *effect if cause.* For prediction, a set of events is represented by a set of sentences *Δ,* and the task is to find the causal consequences of *A* by finding the logical consequences *G* such that *T U Δ = G.* For explanation, events and properties are represented by G, and the task is to find sets of events *A* which could have caused G, in other words, to find *Δ's* such that TUΔ=G. The same theory *T* is used for both prediction and explanation.

In combination with the assumption that all properties which are known to hold must be explained by events, the abductive approach deals correctly with default persistence. Suppose we are told that a property *p* holds at time *t1.* In order to apply default persistence to conclude that it still holds at a later time t2*,* we postulate through abduction the occurrence of an event *e* before t_1 which initiates *p.* In other words, it is necessary to *explain* why *p* holds at t_1. Then default persistence can be applied to show that the property *p* persists from the time of *e* through t_1 and through t_2.

This paper presents the abductive approach to explanation and shows how it deals with default persistence. To illustrate this approach 1 introduce a shortened form of the Event Calculus of Kowalski and Sergot [1986], which is similar to that presented in [Kowalski, 1986J. To demonstrate its practical realisability, 1 describe an abductive mechanism which is related to the techniques of Finger and Genesereth f [1985] and Cox and Pietrzykowski [1986], and is a simplification of the mechanism described by Eshghi [1988], tailored for the shortened form of the Event Calculus.

2. The Invent Calculus

In Kowalski and Sergot's Event Calculus |1986] and its variants [Kowalski, 1986], the ontological primitives are *events,* which initiate and terminate periods during which *properties* hold. The Horn clause subset of the Predicate Calculus is used, augmented with negation-as-failure. The Event Calculus used in this paper is a simplified version of that given by Kowalski and Sergot [1986]. Only two clauses are necessary, as follows.

holds-at(P,T) if (1.1)
happens(E) and E <T and
initiates(E,P) and not clipped(E,P,T)

clipped(E,P,T) if (1.2)
happens(E') and terminates(E\P) and
not T <E' and not E' < E

The formula *holds-at(P,T)* represents that property *P* holds at time *T.* The formula *happens(E)* represents that the event *E* occurs. The time of event *E* is named by the term *time(E).* Times are ordered by the usual comparative operators, but for brevity I will sometimes write *E* instead of *time(E)* in expressions involving temporal ordering. The formula *initiates(E,P)* represents that the event *E* initiates a period during which property *P* holds, and *terminates(E,P)* represents that the event *E* terminates any ongoing period during which property *P* holds. The *not* operator is interpreted as negation-as-failure. The use of negation-as-failure in Axiom (1.1) gives a form of default persistence.

The formula *cligped(E,P,T)* represents that there is a possible mapping of events onto time points in which the property *P* ceases to hold at some time between event *E* and time *T.* The use of negation-as-failure in the definition of *clipped* ensures that *holds-at* works correctly even when events and times are only partially ordered and this mapping is not fully known.

Part of the domain theory is captured in a set of *initiates* and *terminates* clauses. For example, the Blocks World is described by the following clauses. The term *on(X,Y)* names the property that block *X* is on top of block *Y* or at location *Y,* and the term *clear(X)* names the property that block or location *X* has nothing on top of it. The term *move(X,Y)* names the event or act type of moving block *X* onto block or location *Y.*

initiated (E,on(X,Y)) if act(E,move(X,Y)) (2.1)
initiates(E,clear(Z)) if (2.2)
act(E,move(X,Y)) and
holds-at(on(X,Z),time(E)) and Z#Y

terminates(E,clear(Y)) if act(E,move(X,Y)) (2.3)
terminates(E,on(X,Z)) if (2.4)
act(E,move(X,Y)) and Z#Y

To simplify examples, these clauses do not account for the preconditions of events, such as the need for *X* to be clear if *move(X,Y)* is going to have any effect. If necessary, preconditions can easily be incorporated by adding extra conditions to the bodies of *in7itiates* and *terminates* clauses, or can be expressed as integrity constraints [Eshghi, 1988).

The importance of supplying a clear semantics for formulations of default persistence has been demonstrated by Hanks and McDermott [1987]. The example here has a clear semantics because Axioms (1.1) to (2.4) are stratified and therefore have a unique standard model [Apt *et al.,* 1988], [Przymusinski, 1988]. But note that Axiom (2.2) has a *holds-at* in its body. If a *terminates* clause had a *holds-at* in its body, then we would no longer have stratification, because *holds-at* is defined in terms of *terminates* via a negation, and *terminates* would be defined in terms of *holds-at.* Such cases are quite likely to arise. Intuitively, it is clear that this does not cause a problem because of the partial ordering of the events. To show this formally, we need to perform a construction which I will sketch briefly. Each *terminates* clause defined in terms of *holds-at* is folded up with Axioms (1.1) and (1.2) giving a replacement clause of the form

holds-at(P,T2)if (3.1)
Tl <T2and.. . and not holds-at(P'Jl)

This can be replaced by a set of clauses, one for each pair of times t_1,t_2 in the Herbrand universe such that $t_1 < t_2$, of the form

holds-at(P,t2) if. . . and not holds-at(P',tj) (3.2)

Since times are ordered, any set of such clauses is locally stratified [Przymusinski, 1988], and accordingly has a clear semantics. So, for example, the Yale shooting problem can be formulated by a simple set of *initiates* and *terminates* clauses, without the attendant semantic problems described by Hanks and McDermott [1987].

In fact, using negation-as-failure, the correct handling of *not holds-at* in all cases requires some extensions, since *holds-at* can fail simply because the ordering of events and times is not known. There is a distinction between *necessarily-holds-at,* meaning that *holds-at* is true in all possible orderings of times and events, and *possibly-holds-at,* meaning *holds-at* is true in some possible ordering of times and events. Likewise there is a distinction between *possibly-clipped* and *necessarily-clipped.* The existing definitions are for *necessarily-holds-at* in terms of *not possibly-clipped.* But a symmetrical definition is required for *possibly-ho lds-at* in terms of *not necessarily-clipped.* Then, we write *not possibly-holds-at* where we would previously have written *not holds-at,* meaning that *holds-at* fails in all possible orderings of events. To make all this clear would require considerable further discussion, and to incorporate the extensions in this paper would only make the examples more confusing, so I won't mention the mutter again.

3. Prediction and Explanation

The Event Calculus as described can be used to solve prediction problems, that is problems of reasoning from causes to effects, through deduction. The domain is captured by a theory *T* which includes a set of *initiates* and *terminates* clauses and other causal laws as well as the Event Calculus Axioms (1.1) and (1.2). A particular history of events is represented by a set *A* of *happens* and temporal ordering clauses. Then, the properties which hold as a consequence of these events are represented by the set *G* of atomic *holds-at* clauses which are logical consequences of *T U A.* In other words prediction is determining members of *G* where *T U A G.*

The domain theory *T* is strictly causal in the sense none of its rules is of the form *cause if effect.* The intuitive and correct way to express the relationship between causes and effects is with the implication the other way around. Rules of the form *cause if effect,* like those used in Mycin, are almost invariably false, since a given effect usually has many potential causes. Only in particular domains is it possible to assume that there is a unique cause for a given effect, and even then expressing causal laws as inverted implications is counter-intuitive.

However, this begs the question of how explanation, that is reasoning from effects to causes, is to be done. It is tempting to add further clauses to facilitate explanation, possibly of the *cause if effect* form criticised above. But this temptation should be resisted. If the theory *T* adequately captures the relationship between causes and effects it should be equally good for both prediction and explanation. It is important to recognise that explanation can be done through abduction with the same theory. Suppose we are given the theory *T* and we wish to find possible histories of events *A* which would explain a set of properties *G* expressed as *holds-at* clauses. Then we wish to find *Δ's* such that $T \cup \Delta \models G$, and this is abduction.

We can be a bit more precise about what sorts of *A* constitute good explanations. First, *A* should describe a history of events. So it should contain only atomic *happens, act* and

temporal ordering clauses. Second, it should be minimal in the sense that there should not be a Δ^* such that Δ^* A and $T \cup \Delta^* \models$ G. There can, of course, be many minimal $\Delta's$. A third criterion for a good explanation is that it should postulate the fewest events possible. This suggests a preference relation on $\Delta's$ such that Δl is preferable to $A2$ if it contains fewer *happens* clauses. Of course, there may still be many equally preferable minimal As. Clauses which appear in all $\Delta's$ for a given G can be thought of as the defeasibly necessary conditions for G. They are only defeasibly necessary since the addition of new causal laws to T could render G explicable in other ways. Each separate A is a set of defeasibly sufficient conditions for G. They are only defeasibly sufficient, because of default persistence — the addition of further events to A could mean that G is no longer explicable by A

4. Persistence

This section shows how default persistence is handled by the abductive approach to explanation. Suppose we are told that property p holds at time $t1$. In the absence of any further information, what inferences may we reasonably make about a time $t2$ after t_1? The usual notion of default persistence which licenses the inference that P still holds at $t2$, and which is built in to the Event Calculus as well as many other formalisms, is based on two epistemological assumptions and one metaphysical assumption. First, it is assumed that no events occur other than those which are known to occur. Second, it is assumed that no types of event can affect a given property other than those which are known to do so. Third, it is assumed that properties do in fact persist until something happens which affects them.

Incorporated into the framework presented here is a fourth assumption; that every property which is known to hold has an explanation in terms of events. The conclusion that p holds at t_2 is derived partly through deduction and partly through abduction. An event is postulated to explain why p holds at t_1, which initiates p and which occurs before f/, and then default persistence is applied to conclude that p still holds at $t2$. Suppose that the domain theory T comprises Axioms (1.1) to (2.4), that we have a set of axioms A which represents a history of events, and that we are told that block a is at location x at time t_1. So we have

$$holds\text{-}at(on(a,x),t1) \qquad (4,1)$$

This fact is not added directly to the set of axioms A and used to predict new consequences G such that $T \cup \Delta \models$ G. Rather, since it is a *holds-at* fact, it requires explanation. So it is added to the set of theorems G, and suitable As must be sought through abduction which rebalance the sequent $T \cup \Delta \models G$. We do not wish to extend the domain theory, so A must contain only *happens, act* and temporal ordering axioms. For this example all such $\Delta's$ include three axioms of the following form.

$$happens(e) \qquad (5.1)$$
$$act(e,move(a,x)) \qquad (5.2)$$
$$e<t1 \qquad (5.3)$$

In the absence of further axioms, these plus Axioms (1.1) to (2.2) allow us to conclude the default persistence of the property $on(a,x)$ through time $t1$ and through any time $t2$ after $t1$. The new constant e is invented by abduction to name the event it has postulated. The only thing known about the time of this event is that it is before t_1. If such an event were already a part of A then it would not of course be necessary to add anything to A

Suppose that in addition to (4.1), we are also told that the block a is at location y at time t3 which is after $t1$. So we have

$$holds\text{-}at(on(a,x),t_1) \qquad (4.1)$$
$$holds\text{-}at(on(a,y),t_3) \qquad (6.1)$$
$$t_1 < t_2 < t_3 \qquad (6.2)$$

Let us be clear how default persistence should behave with this information. In general, if we are told that a property holds at a time t1, we assume that it still holds at any later time $t2$ unless we have reason to believe that it changes some time between t1 and t2. But in this case, we know that at some time between t1 and r? the block ceases to be at location x and starts to be at location y. In fact, since we do not know when between $t1$ and $t3$ this change occurs, it is not reasonable to conclude anything about whether the book is on the table or the shelf at any given point between these times. This problem is analogous to Kautz's "stolen car" problem [Kautz, 1986], and many approaches to default persistence do not deal with it correctly. For example, with Shoham's logic [Shoham, 19881, default persistence postpones change until as late as possible, and it is then a logical consequence of the information in (4.1), (6.1) and (6.2) that the block is still at location x immediately before time t3.

The approach to default persistence proposed here does not suffer from this problem because of its insistence that every property that holds has an explanation in terms of events. Others have proposed similar solutions using deduction [Morgenstern and Stein, 1988], [Lifschitz and Rabinov, 1988]. But using abduction, rather than adding (4.1) and (6.1) to the set of axioms Δ, they are added to the set of theorems G. This leads to the rebalancing of the sequent $T \cup \Delta$ G via the abduction of axioms (5.1) to (5.3) to explain (4.1) as described above, and also the abduction of the following four axioms to explain (6.1).

$$happens(e') \qquad (7.1)$$
$$act(e',move(a,y)) \qquad (7.2)$$
$$e' < t_3 \qquad (7.3)$$
$$t_1 < e' \qquad (7.4)$$

With the addition of (7.1) to (7.4) to Δ, because the relative ordering of e' and t2 is not known, default persistence no longer licenses the conclusion that $holds\text{-}at(on(ajc),t2)$. Axioms (7.1) to (7.4) will be present in any A which explains G, and can be thought of as the necessary conditions for G given T. A more complicated example might yield many $\Delta's$, and each such A is a set of (defeasibly) sufficient conditions for G given T.

Unlike many formulations of persistence, that presented here works forwards only. Suppose again that we are told that property p holds at time $t1$. In the absence of any further information, what inferences may we reasonably make about a time to before $t1$? The three assumptions which license the default inference that p still holds at a time $t2$ after $t1$ do not apply to a time before f/. With the additional assumption that properties require explanations, we conclude that some event must have occurred to initiate/?. But we have no idea when that event occurred — it may have been before or after to. So there is no reason to suppose that p holds at to. The correct way to deal with persistence is to ensure that it works forwards only.

5. The Abductive Mechanism

The abductive approach to explanation can be realised using a mechanism which is a straightforward extension of SLD resolution. Let us consider SLD resolution first. Given a set of definite clauses *T* and a goal clause <—Go, an SLD refutation of *<-G0* is a sequence of goal clauses *<—Go...<—G_n* where *<-G_n* is the empty clause and each <— Gi+1 is obtained from <— *Gi* by resolving one of its literals with one of the clauses in *T.* In a Prolog interpreter, the leftmost literal is always selected. Since there may be many clauses in *T* which can be resolved with the selected literal, a space of possible refutations is defined, which may be searched, for example, depth-first by a simple chronological backtracking procedure. Now suppose that there is some *-G, whose selected literal *g* will not resolve with any clause in T. Usually this means that sequences beginning with <— *Go* . . . <— Gi are not worth exploring. But if we are searching for a set of unit clauses *A* such that *TUΔ⊨G0,* then clearly by letting *A* include a unit clause which resolves with *g,* we can continue the search with <— Gi+1 equal to <—G minus the literal *g.* This suggests the following extension to SLD resolution.

A subset of the predicate symbols mentioned in *T* are designated as the *abducibles.* A literal whose predicate symbol is abducible is also called abducible. To find a set of unit clauses Δ_n such that $TU\Delta_n \models G0$ and A_n mentions only abducibles, a refutation of the form *<—G0,ΔO* . . . *<G_n,*Δ_n is constructed, where each <—Gi is a goal clause, each *A;* is a set of unit clauses mentioning only abducibles, <— G_n is the empty clause, *Δ0* is the empty set, and each *<—Gi+1,Δi+1* is obtained from *<Gi,Δi* as follows. If *g,* the selected literal of <—Gi, can be resolved with one of the clauses in T, then a single resolution step is taken as described above and *Δi+1* is *Δi.* If *g* is abducible and cannot be resolved with any clause in T, then *Gi+1* is *Gi* minus *g* and *Δi +1* is *Ai* plus the unit clause *g'*— where *g'* is *g* with all its variables replaced by skolem constants [Cox and Pietrzykowski, 1986]. If *g* were not skolemised, all the variables in g'→ would be universally quantified, which would make it unnecessarily strong. Its variables only need to be existentially quantified for it be resolvable with *g.* The accumulated set of unit clauses Δ_n is called the *residue.*

The basic mechanism can be extended to cope with negation-as-failure (Eshghi and Kowalski [1988] and Poole [1988) discuss the use of abduction as a general framework for default reasoning). This is essential to cope with default persistence in the Event Calculus. Suppose that the selected literal of the current goal clause is *not g.* The usual negation-as-failure method is adopted, and *not g* is assumed to be true if *g* cannot be proved with the current residue. But later in the refutation, additions to the residue can make *g* provable. Accordingly, it is necessary to record all negated assumptions, and whenever new clauses are added to the residue, these assumptions must be rechecked. This is a potential computational bottleneck, but some form of incremental mechanism could be used to minimise this [Sadri and Kowalski, 1988], [Shanahan, 1987]. The negated assumptions that are recorded can be thought of as part of the residue, and rechecking them is like checking for consistency with an implicit integrity constraint. As with abducible literals, all the variables in a recorded negated assumption are replaced by skolem constants.

A further complication arises with nested negation-as-failure. Suppose that there is a clause of the form *g <— not h'* and that *h'* is not provable with the current residue. Then an attempt to prove *not g* using SLD resolution with negation-as-failure will fail because it is not possible to prove h'. Yet it might have been possible to render *h'* provable by adding further clauses to the residue. So rather than using SLD resolution to try to show *h',* abduction is used instead and is allowed to add to the residue. This procedure can be generalised to any level of nesting — SLD is used at even levels and abduction is used at odd levels.

This general abductive mechanism can be specialised for the Event Calculus axioms above. Any goal of the form *happens (E), act(EA)* and *T1 < T2* is abducible. The initial goal clause is of the form *<-holds-at(P_1,T_1).* . ., *holds-at(P_n,T_n),* and the procedure is then the same as above. Of course, a complete search space for a given *G* may contain many Δ's, as indeed there may be many possible explanations for G. By ordering the branches of the search space appropriately, the simplest explanations — those which postulate the fewest events — will be generated first. One heuristic for extracting the simplest explanations first is to reuse old skolem constants rather than generating new ones. For example, if the residue contains *act(s,move(a,b)),* and the goal clause is *<—act(E,move(a,b)),*. . . , where *s* is a skolem constant, then the simplest way of resolving away the *act* literal is just to bind *E* to *s,* rather than to postulate another event and add another *act* clause to the residue. Later on though, this binding may lead to a failing branch of the search, in which case backtracking takes place and a new event has to be postulated after all. A similar case arises if a skolem constant has already been created, but can be eliminated later. For example, suppose the residue contains *act(e,move(s,b))* and the goal clause is *<-act(e,move(a,b)).* The simplest way to resolve away the *act* literal this time is to replace all occurrences of the skolem constant *s* by *a,* rather than adding a new *act* clause to the residue. Again, later failure may mean that backtracking undoes this decision. In general, explanations can be generated in order of simplicity by abandoning a depth-first search strategy in favour of one which explores branches which don't postulate new events first.

Let us consider a trivial example of this mechanism applied

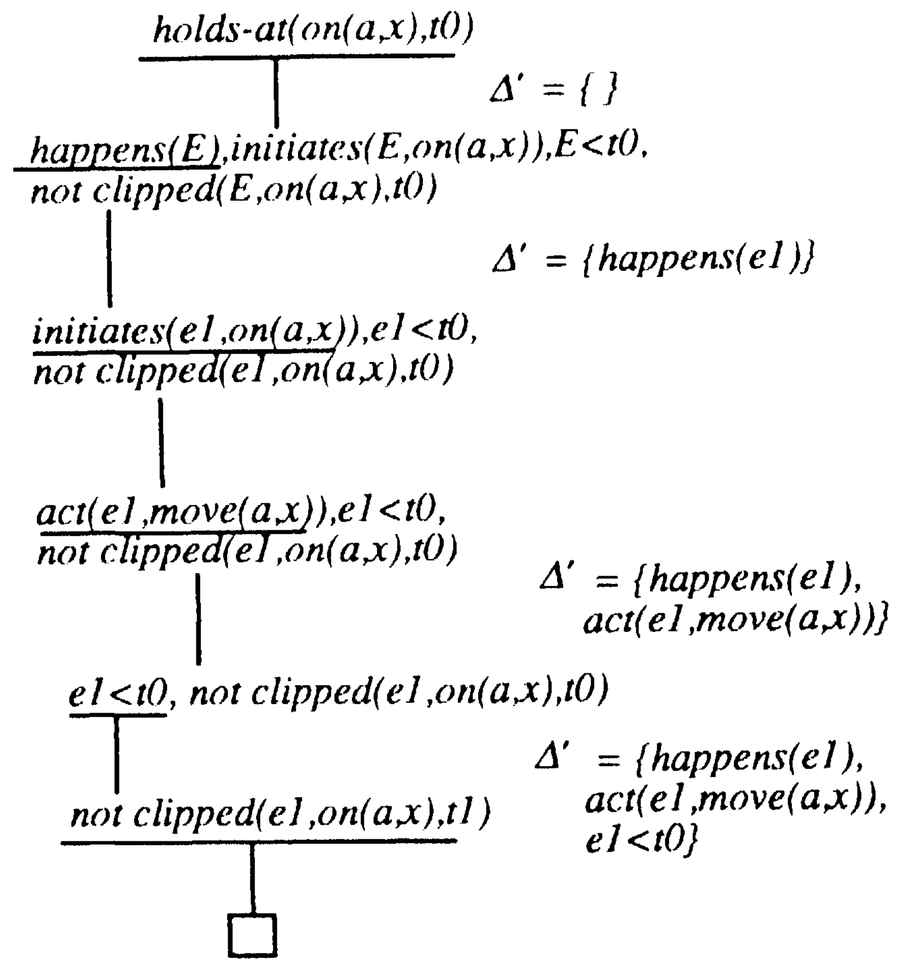

Figure 1.

to explanation. Given Axioms (2.2) to (2.4) for the Blocks World, suppose that we require an explanation for the fact that *holds-at(on(a,x),tO).* The search space for this example is shown in Figure 1. Abduction generates the residue *Δ'={happens(el), act(el,move(ajc)), el<t0).*

The next example is more complicated and illustates most of the features of the mechanism I have described. Suppose that we are given that *t0<t1* and *tl<t2,* and we want an explanation for *holds-at(on(a,x),tO) and holds-at(on(a,x),t2) and holds-at(clear(x),tl).* This is an extension of the previous example, and the search space in Figure 2 would be appended to the one above if the extra goals were added. It is assumed that the residue already contains *Δ',* and the overall residue is *Δ = Δ' U Δ".*

When abducing an event to explain *holds-at(on(a,x),t2),* the mechanism has the option of supposing that it is the same event as the one it has already postulated *el,* or of postulating a new event *e2.* This gives rise to two branches in the search space. Furthermore, *el* does initiate the property *on(a,x)* and does occur before r2, and it cannot yet be shown that *clipped(el,on(a,x),t2).* But when, in order to explain *holds-at(clear(x),tl),* an event *e3* has to be postulated which initiates *clear(x),* clauses are added to the residue which make it possible to show that *clipped(el,on(a,x),t2),* and this gives rise to a failure. The mechanism backtracks and explores the second branch of the search space, which succeeds with the overall residue *A = {happens(el), act(el,move(a,x)), el<tO, happens(e2), act(e2,move(a,x)), e2<t2, happens(e3), act(e3,move(a,l)), e3<tl, e3<e2, el<e3}.* The skolem constant / represents an unspecified location, and could later be replaced by the name of a real location. Note that if the goals had been presented in a different order, then the first branch might not have been explored. Also, if the goal *holds-at(clear(x),tl)* were not included then the first branch of the search space would succeed with the simplest explanation, postulating only the event *el* to explain both of the other *holds-at* goals. The solution of the last *not clipped* goal shows how extra constraints on temporal ordering can be generated even within a negation. Without the addition of the clause *el<e2,* it would have been possible to prove *clipped(e3,clear(x),t!l*

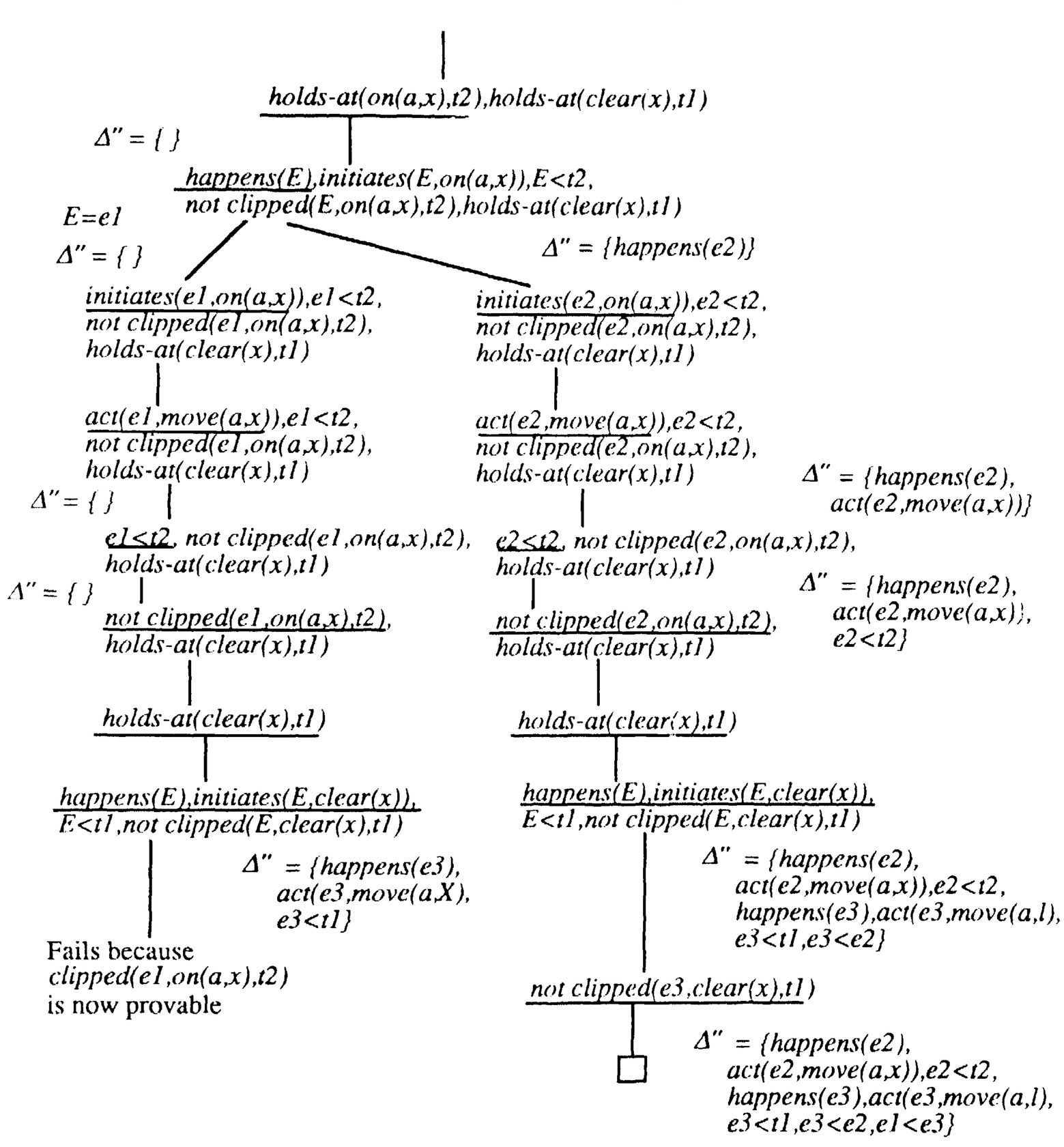

Figure 2.

6. Concluding Remarks

Finger and Genesereth [19851 describe an extension to resolution which is similar to the mechanism presented here, but have applied it to design synthesis rather than temporal reasoning. Cox and Pietrzykowski [1986] also describe a

related technique. Eshghi [1988] has applied abduction to temporal reasoning, specifically to planning, using a form of Kowalski and Sergot's Event Calculus which is very different from their original formulation. His approach employs meta-level integrity constraints to represent preconditions for actions as well as to handle default persistence, and uses an elaborate mechanism to cope with explicit equalities which are generated in place of the usual implicit bindings generated by a resolution system.

The approach taken in this paper is to use stratification semantics for negation-as-failure, and to use negation-as-failure to give default persistence. Abduction is used only for explanation. Eshghi and Kowalski [1988], however, present an abduction semantics for negation-as-failure itself, and Poole [1988] also presents an abductive framework for default reasoning. This suggests that both persistence and explanation could be done in a purely abductive framework, but this possibility needs further investigation.

Morgenstem and Stein [1988] and Lifschitz and Rabinov [1988] tackle a similar problem to the one addressed in this paper, the former using model preference and the latter using circumscription. The relationship between the three approaches is not yet clear and warrants further study.

A prototype of the system described has been implemented in Prolog. This has highlighted the need for a more sophisticated control strategy than that provided by simple chronological backtracking, since the system spends much time exploring possible explanations which are clearly ridiculous, and often loops in subtle and unexpected ways.

Acknowledgements

Thanks for discussion and inspiration to Kave Eshghi, Marek Sergot, Sury Sripada, Vladimir Lifschitz, Bob Kowalski and Chris Evans.

References

[Apt *et al.,* 1988] Apt K., Blair H. and Walker A., Towards a Declarative Theory of Knowledge, in *Foundations of Deductive Databases and Logic Programming,* ed Minker J., Morgan Kaufman (1988), p 89.

[Cox and Pietrzykowski, 1986] Cox P.T. and Pietrzykowski T., Causes for Events: Their Computation and Applications, *Proceedings CADE 86,* p 608.

[Eshghi, 1988] Eshghi K., Abductive Planning with Event Calculus, *Proceedings 5th International Conference on Logic Programming* (1988), p 562.

[Eshghi and Kowalski, 1988] Eshghi K. and Kowalski R.A., Abduction Compared with Negation by Failure, Imperial College Department of Computing Technical Report (1988), to appear in *Proceedings 6th International Conference on Logic Programming* (1989).

[Finger and Genesereth, 1985] Finger J.J. and Genesereth M.R., RESIDUE: A Deductive Approach to Design Synthesis, Stanford University Technical Report no. CS-85-1035. (1985).

[Hanks and McDermott, 1987] Hanks S. and McDermott D., Nonmonotonic Logic and Temporal Projection, *Artificial Intelligence,* vol 33 (1987), p 379.

[Kautz, 1986] Kautz H., The Logic of Persistence, *Proceedings AAAI 86,* p401.

[Kowalski, 1986] Kowalski R.A., Database Updates in the Event Calculus, Imperial College Department of Computing Technical Report no. DOC 86/12 (1986).

[Kowalski and Sergot, 1986] Kowalski R.A. and Sergot M., A Logic-Based Calculus of Events, *New Generation Computing,* vol 4 (1986), p 67.

[Lifschitz and Rabinov, 1988] Lifschitz V. and Rabinov A., Miracles in Formal Theories of Action, Stanford University Technical Report (1988), to appear in *Artificial Intelligence.*

[Morgenstem and Stein, 1988] Morgenstem L. and Stein L.A., Why Things Go Wrong: A Formal Theory of Causal Reasoning, *Proceedings AAAI 88,* p 518.

[Poole, 1988] Poole D.L., A Logical Framework for Default Reasoning, *Artificial Intelligence,* vol 36 (1988), p 27.

[Przymusinski, 1988] Przymusinski T., On the Declarative Semantics of Deductive Databases and Logic Programs, in *Foundations of Deductive Databases and Logic Programming,* ed Minker J., Morgan Kaufman (1988), p 193.

[Sadri and Kowalski, 1988] Sadri F. and Kowalski R.A., A Theorem Proving Approach to Database Integrity, in *Foundations of Deductive Databases and Logic Programming,* ed Minker J., Morgan Kaufman (1988), p 313.

[Shanahan, 1987] Shanahan M.P., An Incremental Theorem Prover, *Proceedings IJCAI 87,* p 987.

[Shoham, 1988] Shoham Y., Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence, MIT Press (1988).

A knowledge-level account of abduction
(preliminary version)

Hector J. Levesque*
Dept. of Computer Science
University of Toronto
Toronto, Canada M5S 1A4

Abstract

In this paper, we consider a new definition of abduction that makes it depend on an underlying formal model of belief. In particular, different models of belief will give rise to different forms of abductive reasoning. Based on this definition, we then prove three main theorems: first, that when belief is closed under logical implication, the corresponding form of abduction is precisely what is performed by the ATMS as characterized by Reiter and de Kleer; second, that with the more limited "explicit" belief defined by Levesque, the required abduction is computationally tractable in certain cases where the ATMS is not; and finally, that something is believed in the implicit sense iff repeatedly applying a limited abduction operator eventually yields something that is believed in the explicit sense. This last result relates deduction and abduction as well as limited and unlimited reasoning all within the context of a logic of belief.

1 Introduction

Using the terminology of C. S. Peirce, given sentences α, β, and $(\alpha \supset \beta)$, there are three operations one can consider: from α and $(\alpha \supset \beta)$, one might *deduce* β; from α and β, one might *induce* $(\alpha \supset^1 \beta)$; and from β and $(\alpha \supset \beta)$, one might *abduce* α. Of course, characterizing precisely what should be deduced, induced, or abduced in various circumstances is quite another matter, and the last of these is the subject of this paper.

Abduction can be thought of as a form of hypothetical reasoning. To ask what can be abduced from β is to ask for an α which, in conjunction with background knowledge,[2] is sufficient to account for β. When α and β are about the physical world, this normally involves finding a cause α for an observed effect β. For instance, β might say that a symptom of some sort is observed and α might say that a disease is present. We often say in this case, that α *explains* β. But not all abduction is concerned with cause and effect. If we happen to known that Marc is 3 or 4 years old, the fact that he is not yet 4 does not *explain* his being 3, although it does imply it, given what is known.[3] It would be more accurate to say that the α is sufficient to tell us that the β is true. But this is a bit cumbersome, so with this caveat in mind, we will often use the explanation terminology here.

When it comes to formally characterizing abduction, existing approaches fall into two broad camps: those, like [Reggia, 1983, Allernand *et al.*, 1987], that are *set-cover* based, and those, like [Poole, 1988, Eshghi and Kowalski, 1988], that are *logic* based. In the former case, abduction is defined over sets of observations and hypotheses, in terms of coverings, parsimony, plausibility, and the like. A disadvantage of this approach is that it is difficult to express how a small change in the background knowledge can contribute to changing what counts as an explanation. In the latter case, however, this knowledge is represented directly as a logical theory, and α is considered an explanation for β if (1) it is logically consistent with what is known, and (2) together with this knowledge, logically implies β. The disadvantage of defining abduction in this way is that it locks the specification of reasoning into global properties of the logic such as consistency and implication. Different reasoning abilities, deductive or abductive, will then require different notions of implication or consistency.

Here we take a different approach and characterize abduction in terms of a model of belief. When belief is closed under ordinary logical consequence, this account will coincide with the idealized logic-based version. However, we can look at different forms of abduction by varying the underlying notion of belief, without changing the meaning of implication. This *knowledge-level* approach [Newell, 1982, Levesque and Brachman, 1986] will also

*Fellow of The Canadian Institute for Advanced Research. This research was made possible in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

[1] More likely, one would want to induce $\forall x(\alpha \supset \beta)$ from instances of α and β.

[2] The distinction between knowledge and belief is not important here, and we will use the terms interchangeably.

[3] Another reason for distinguishing this from explanation is that we normally say that a explains β only when we believe β to be true, for example, when we have observed the symptoms in question. So a true account of explanation *per se* is complicated by the fact that it must consider what was known *prior* to believing β [Gardenfors, 1988], or else there will be nothing left to explain, given what is known.

force us to characterize the abduction task independently of how the knowledge is represented, and thus afford the greatest freedom in how to represent and manipulate this knowledge at the symbol level.

In the next section, we introduce notation, discuss the need for a simplicity measure, and define a new knowledge-level operator EXPLAIN. In Section 3, we discuss the concept of regular belief and its relation to EXPLAIN. In Section 4, we examine a form of belief that is closed under logical consequence, and the ATMS as an abductive reasoner. In Section 5, we consider the specification of a more limited abductive reasoner. Finally, we draw conclusions in Section 7.

2 Abduction at the knowledge level

To define abduction, we start with $\mathcal{L}$, a standard propositional language (except that for convenience, we include a special constant $\Box$, for falsity). All beliefs will be expressed in $\mathcal{L}$. We use p, q, and r to range over the propositional letters of $\mathcal{L}$; α, β, and γ to range over the sentences of $\mathcal{L}$; m to range over the literals of $\mathcal{L}$; $\overline{m}$ to mean the complement of m; and, $\bigwedge\{\alpha_i\}$ to conjoin a set of sentences, and $\bigvee\{\alpha_i\}$ to disjoin them.

To talk about what is or is not believed, we use a logical language $\mathcal{L}^*$ that is structured like $\mathcal{L}$, except that all of its atomic sentences are of the form $\mathbf{B}\alpha$, where α is a sentence of $\mathcal{L}$.[4] For different kinds of belief, we use a subscript on the belief operator. So, $\mathbf{B}_\lambda\alpha$ says that α is a belief of type λ.

The languages $\mathcal{L}$ and $\mathcal{L}^*$ are both interpreted in the standard way in that the truth values of non-atomic sentences are the usual functions of the truth values of their components. For the atomic sentences of $\mathcal{L}$, an *assignment* is a total function from the propositional letters to $\{0,1\}$, and $w \models \alpha$ means that α is true with respect to assignment w according to the ordinary truth table (where $\Box$ always comes out false). For the atomic sentences of $\mathcal{L}^*$, we assume that an *epistemic state* of some sort determines which sentences of $\mathcal{L}$ are believed. The notation $e \models \mathbf{B}_\lambda\alpha$ says that $\mathbf{B}_\lambda\alpha$ is true at epistemic state e, which we will take as a primitive notion for now, until we look at specific types of belief.

For any sentence α of $\mathcal{L}$, it will be useful to talk about $\|\alpha\|$, the proposition expressed by α. Nothing hinges on how exactly propositions are defined, but for concreteness, they can be taken to be the set of all assignments where the sentence in question is true. Similarly, $\|\{\alpha_1,\ldots,\alpha_n\}\|$ is defined as $\{\|\alpha_1\|,\ldots,\|\alpha_n\|\}$.

2.1 Simplicity and uniqueness

Deductive and abductive reasoning appear to be duals, but one difference between the two is that in the case of deduction, we are usually interested in *testing* if some sentence is deducible, while in the case of abduction, we want to *produce* a sentence that is abducible.[5]

For example, consider a medical domain where sentences of $\mathcal{L}$ stand for properties that may or may not hold of a certain patient. Suppose we know that *male* and *(hepatitis ⊃ jaundice)* are both true. If we observe jaundice in the patient, we might be interested in determining what might explain it, based on what we know about the patient. In other words, we want to reason abductively from *jaundice,* to find something that accounts for it, given what is known. In this case, the answer is clearly *hepatitis,* but it is not obvious how to characterize in general the answers we are looking for.

First of all, we cannot expect a *single* explanation since, for example,

((¬¬*hepatitis* ∧ *migraines)* ∨ *(hepatitis* ∧ ¬*migraines))*

also accounts for *jaundice.* But even if we factor out logically equivalent sentences and think in terms of propositions, there will be propositions that are logically too strong, and others that are logically too weak. For instance, *(hepatitis* ∧ *migraines)* accounts for the jaundice in that it is consistent with what is known, and if it were true, then *jaundice* would be too. Similarly, *(hepatitis* ∨¬*male)* accounts for *jaundice* since it too is consistent with what is known, and if it were true, then *jaundice* would be also, since *male* is known to be true. Yet *(hepatitis/\ migraines)* implies *hepatitis* which implies *(hepatitis* ∨ -¬*malc).*

So what is it that distinguishes || *hepatitis*\\ from these other propositions? Is there a way to sort this out purely logically (in terms of sets of possible worlds and distance measures or whatever) and define an appropriate explanation? As it turns out, the answer is no. To see this, suppose to the contrary that there were a function *F* that given the proposition expressed by *( )* and the one expressed by B would always return the one expressed by *a.* That is, suppose that for every a and β, $F(\|(\alpha \supset \beta)\|,\|\beta\|)=\|\alpha\|$. Then, we would have $F(\|(q \supset q)\|,\|q\|)=\|q\|$ and $F(\|(\Box \supset q)\|,\|q\|)=\|\Box\|$. However, $\|(q \supset q)\|=\|(\Box \supset q)\|$ since the two sentences are logically equivalent.[6] But this implies that $\|\Box\|=\|q\|$, which is incorrect. So such a function *F* cannot exist, and we are forced to go beyond the logic of the sentences (that is, beyond the propositions expressed) to differentiate *hepatitis* from other potential explanations.

One obvious approach is to maintain a list of sentences that are marked as possible hypotheses as is done in [Poole, 1988, Reiter, 1987] and to only consider sentences appearing in this list. But this fails to account for why we find *hepatitis* so compelling as the unique explanation for *jaundice* in the above. Perhaps it is because *hepatitis* does not deal with any other conditions, either to insist on (conjoin) irrelevant restrictions like *migraines,* or to allow for (disjoin) possibilities known to be false like *-imalc.* This suggests that we should be looking for sentences that are as *simple* as possible in their subject matter. With this notion of simplicity in mind, we are ready to provide a formal definition of abduction in terms of belief.

[4] For the purpose of this paper, therefore, we will not consider beliefs about other beliefs.

[5] However, see Section 7 where the symmetry between deduction and abduction is reconsidered.

[6] A stronger argument would be needed for a notion of proposition that was finer-grained than logical equivalence.

2.2 A general definition

First, we define explanation wrt an epistemic state e for a type of belief λ:

Definition 1 α **expl**$_\lambda$ β **wrt** e **iff** $e \models [\mathbf{B}_\lambda(\alpha \supset \beta) \wedge \neg \mathbf{B}_\lambda \neg \alpha]$.[7]

This definition does not distinguish between trivial and non-trivial occurrences of $B(\alpha \supset \beta)$. For example, assuming that *(jaundice* $\supset$ *jaundice)* is believed but that $\neg$*jaundice* is not, *jaundice expl*$_\lambda$ *jaundice* wrt *e* holds, that is, having jaundice is clearly (and trivially) sufficient to account for having jaundice. More generally, if nothing is believed about β other than logical truths, then there will only be trivial explanations. In addition, for many types of belief, we have that if β is believed, then there will be no explanations at all, whereas ifβ itself is believed, then $\neg\Box$ will be-the unique explanation.

As discussed above, the definition of explanation must depend on some syntactic criterion of simplicity. Perhaps the easiest one is the following:

Definition 2 The literals of α, LITS(α), is defined by:
LITS($\Box$)=$\emptyset$; LITS(p)=$\{p\}$;
LITS($\neg\alpha$)=$\{\overline{m} \mid m \in \text{LITS}(\alpha)\}$;
LITS($\alpha \wedge \beta$)=LITS($\alpha \vee \beta$)=LITS(α) $\cup$ LITS(β).

Definition 3 α is *simpler* than β (written $\alpha \prec \beta$) iff LITS(α) $\subset$ LITS(β); also $\alpha \preceq \beta$ iff LITS(α) $\subseteq$ LITS(β).[8]

So "simpler" means "containing fewer propositional letters," but keeping track of their polarity. For example $(p \wedge \neg q) \prec (q \supset (p \vee r))$, but $(p \wedge \neg q) \not\prec (\neg q \supset (p \vee r))$. We define a simplest explanation in the obvious way:

Definition 4
α **min-expl**$_\lambda$ β **wrt** e iff α **expl**$_\lambda$ β **wrt** e and
for no $\alpha^* \prec \alpha$ is it the case that α^* **expl**$_\lambda$ β **wrt** e.

Finally, since there may be more than one simplest- explanation, and since we do not really care at this level how each simplest explanation is expressed, the task of abduction will be to return the set of propositions of all simplest explanations:

Definition 5
$\text{EXPLAIN}_\lambda[\![e, \beta]\!] = ||\{\alpha \mid \alpha \text{ min-expl}_\lambda \beta \text{ wrt } e\}||$.

These simplest explanations should be understood *disjunctively.* For example, if we know that *(p*$_1$ $\supset$ *q\)* and *(P'2* $\supset$ q_2), then *pi* is a simplest explanation of *(q*$_1$ Vq_2) and so is P2. However, it is the disjunction (p1 VP2) that fully and non-trivially accounts for *(q* V q2).

[7]In the final paper, various other options for these two conjuncts will be examined. Instead of the first one, we might want to say that if we were *told at,* then we would believe *β,* which need not be the same as believing $(\alpha \supset \beta)$ in the presence of defaults; instead of the second one, we might prefer saying for a given 7 that we do *not* believe $(\alpha \supset \gamma)$ (to handle negative evidence), which for regular belief (see below) coincides with the above when 7 is $\Box$

[8]For some applications, we might wish to use a superset of this relation. For example, we might want to say that $p \prec q$ even though both are atomic, if we consider *p* to be much more likely than *q.* But we should never have to consider a *subset* of the relation.

This completes the knowledge-level characterization of abduction. The theorems to follow below (especially the relationship to the ATMS) are the best evidence that the definition is apt. But it is worth noting here how simple and general the account is. It is the first (to my knowledge) that not only works for sentences *β* of arbitrary syntactic form, but is also sensitive to what is known without requiring an explicit list of the known sentences. In other words, it does *not* depend in any way on how the epistemic state e is represented (and so is truly at the knowledge level). Computations at the symbol level, of course, will need to operate on finite symbolic representations of that state. Typically, for each type of belief λ, there will be a function $\Re_\lambda$ that maps (finite) sets of sentences into epistemic states. At the symbol level, there will be a procedure of some sort that takes a representation of knowledge KB and a sentence β as arguments, and produces a set of sentences by abductive reasoning. For an abductive procedure to be *correct,* the sentences it returns must express all and only the simplest explanations of *(3* wrt the epistemic state represented by KB. Thus, what we will want to establish for various types of belief and associated computational procedures explain[KB,β] is the following:[9]

$$\text{EXPLAIN}_\lambda[\![\Re_\lambda(\text{KB}), \beta]\!] = ||\,\textbf{explain}[\text{KB}, \beta]\,||.$$

Note that for this general account, correctness does not require the sentences returned by the symbol-level procedure to be in a certain syntactic form, provided that they express the right propositions.

3 A generic abduction operation

Before looking at two specific types of belief, we define what it means for belief to be *regular.* In what follows, we use the following notation: *x, y,* and *z* stand for *clauses,* that is, finite sets of literals always understood disjunctively; the empty clause is $\Box$; *(x — y)* is the clause whose literals are those in the set difference *of x* and *y; x* is the set of complements of the literals in *x,* now understood conjunctively; E and F stand for sets of clauses; for any Σ, $\mu\Sigma$ is the set of smallest (in the sense of subset) elements of Σ; and finally, CNF*(cx)* is the set of smallest clauses that result from converting a to conjunctive normal form, and analogously for DNF(a).

Definition G A type of belief A is *regular* iff for every epistemic state, the following sentences of £* are true:

1. $\mathbf{B}_\lambda \neg \Box$;
2. $(\mathbf{B}_\lambda \alpha \vee \mathbf{B}_\lambda \beta) \supset \mathbf{B}_\lambda(\alpha \vee \beta)$;
3. $\mathbf{B}_\lambda(\alpha \wedge \beta) \supset (\mathbf{B}_\lambda \alpha \wedge \mathbf{B}_\lambda \beta)$;
4. $(\mathbf{B}_\lambda \alpha \wedge \mathbf{B}_\lambda \beta) \supset \mathbf{B}_\lambda(\alpha \wedge \beta)$;
5. $\mathbf{B}_\lambda \alpha \equiv \mathbf{B}_\lambda \alpha^*$, if α^* is α in CNF or DNF, or is the result of replacing any subformula β in α by β^*, where (recursively) $\mathbf{B}_\lambda \beta \equiv \mathbf{B}_\lambda \beta^*$ is always true.[10]

We now define a very general operation on two sets of clauses (which we will eventually use for both types of belief below) as follows:

[9]We use this font to indicate a symbol level procedure.
[10]Note that this does *not* sanction replacing *β* by everything logically equivalent to it.

Definition 7 $\nabla(\Sigma,\Gamma) =$

$$\mu\left\{\overline{z} \mid \begin{array}{l}\forall y \in \Sigma,\ y \not\subseteq z \quad \text{and} \\ \forall x \in \Gamma,\ \exists y \in \Sigma,\ x \cap y \neq \emptyset \text{ and } (y - x) \subseteq z\end{array}\right\}$$

This generalizes the MIN-SUPPORTS operation of [Reiter and de Kleer, 1987]: MIN-SUPPORTS(x,Σ) = $\nabla(\text{IMPS}(\Sigma), \{x\})$, where IMPS($\Sigma$) is defined below. Informally, the elements of Σ should be thought of as the clauses that are believed, the elements of Γ as the clauses to be explained, and $\nabla(\Sigma,\Gamma)$ as the minimal explanations. For instance, if

$\Sigma = \{(p_1 \vee \overline{p}_4), (p_1 \vee \overline{p}_5 \vee p_7), (p_2 \vee \overline{p}_6 \vee p_7), (\overline{p}_3 \vee \overline{p}_8), (\overline{p}_4 \vee \overline{p}_8)\}$
and
$\Gamma = \{p_1, (p_2 \vee \overline{p}_3)\}$,

then $\nabla(\Sigma,\Gamma) = \{(p_4 \wedge p_6 \wedge \overline{p}_7), (p_5 \wedge p_6 \wedge \overline{p}_7), (p_5 \wedge \overline{p}_7 \wedge p_8)\}$. That is, if we take one of these explanations, $(p_4 \wedge p_6 \wedge \overline{p}_7)$, and assume that it and the elements of Σ are true, we get that $(p_1 \wedge p_2)$ must be true, which implies that the clauses of T must all be true. The other two explanations work analogously. Note that $(p_4 \wedge p_8)$ is *not* returned as an explanation since it is believed to be false, that is, its negation is an element of Σ.

The important property of ∇ is that although it only deals with clauses, it can be used to provide correct abductive reasoning for regular belief:

Theorem 1 *For regular belief,*
$\text{EXPLAIN}_\lambda[\![e,\beta]\!] = \|\nabla(\{y \mid e \models \mathbf{B}_\lambda y\}, \text{CNF}(\beta))\|$.

Proof: The proof depends on two key lemmas:

Lemma 1.1 *If* α expl_λ β **wrt** e *then*
$\exists \overline{z} \in \text{DNF}(\alpha)$, $\overline{z}$ expl_λ β **wrt** e.

Lemma 1.2 $\overline{x}$ expl_λ β **wrt** e *iff*
$\exists z \subseteq x, \overline{z} \in \nabla(\{y \mid e \models \mathbf{B}_\lambda y\}, \text{CNF}(\beta))$.

The final paper proves these and the theorem. ■

What this theorem establishes at a very abstract level is that for regular belief, it is sufficient to work with the set of clauses believed and the CNF of the sentence to be explained. This will immediately lead to two abductive procedures below.

4 Case 1: Implicit belief

The first notion of belief we consider is the "classical" one where beliefs are closed under logical consequence. Following [Levesque, 1984], we call this *implicit belief* and use B_1 as the belief operator. An epistemic state for implicit belief can be modeled by any set of assignments, where we have the following:

$$e \models \mathbf{B}_1\alpha \quad \text{iff} \quad \text{for every } w \in e,\ w \models \alpha.$$

If KB is a set of sentences, then R_1(KB), the epistemic state represented by KB, is modeled by the set of all assignments that satisfy every element of KB. What is believed in this state is precisely what follows from KB, that is, if $e = \Re_1(\text{KB})$, we have that $e \models \mathbf{B}_1\alpha$ iff KB $\models \alpha$. From this it follows that

α expl_1 β **wrt** e iff
KB $\cup \{\alpha\} \models \beta$ and KB U {a} is consistent,

which is precisely the account of explanation given by (among others) Poole in [Poole, 1988].

4.1 The ATMS

One abductive procedure that is receiving considerable attention is the ATMS [de Kleer, 1986]. Unfortunately, descriptions of the overall function computed by the ATMS have been largely in terms of *how* it goes about computing it. The first account that attempted to provide a logical reconstruction was that of Reiter and de Kleer in [Reiter and de Kleer, 1987]. Although idiosyncratic terms like labels, nodes, and nogoods are no longer part of the formulation, their definition is in terms of clause intersections and differences, notions that are (arguably) still best understood as symbol level manipulations of sentences in a certain form. However, given their characterization, they are able to show the following:

Definition 8
The *implicants* of Σ, $\text{IMPS}(\Sigma) = \{y \mid \Sigma \models y\}$.

Theorem 2 (Reiter and de Kleer) *Given a set of Horn clauses* Σ *and a letter* p, *the ATMS procedure is defined by* $\mathbf{atms}[\Sigma,p] = \{(q_1 \wedge \ldots \wedge q_k) \mid k \geq 0$ and $\{\overline{q}_1, \ldots, \overline{q}_k, p\} \in \mu\text{IMPS}(\Sigma)\}$.

In fact, Reiter and de Kleer generalize the account of the ATMS to where the first argument is not necessarily Horn and the second argument is any clause. However, we can go even further by noting that

$$\begin{aligned}\mathbf{atms}[\Sigma,p] &= \{(\overline{y} - \{\overline{p}\}) \mid p \in y \text{ and } y \in \mu\text{IMPS}(\Sigma)\} \\ &= \mu\{\overline{z} \mid \forall y \in \text{IMPS}(\Sigma),\ y \not\subseteq z \quad \text{and} \\ &\qquad \exists y \in \text{IMPS}(\Sigma),\ p \in y \text{ and } (y - \{p\}) = z\} \\ &= \nabla(\text{IMPS}(\Sigma), \{\{p\}\}).\end{aligned}$$

Using this as a pattern, we can define a generalized ATMS as follows:[11]

Definition 9 $\mathbf{gatms}[\Sigma,\beta] = \nabla(\text{IMPS}(\Sigma), \text{CNF}(\beta))$.

Clearly this coincides with the ATMS specification when Σ is a set of Horn clauses and β is a propositional letter. But what do these operations *mean,* and why should anyone care about them? The answer, we claim, is that the ATMS procedure correctly performs abduction for implicit belief:

Theorem 3 $\text{EXPLAIN}_1[\![\Re_1(\Sigma),\beta]\!] = \|\mathbf{gatms}[\Sigma,\beta]\|$.

Proof: It is not hard to show that implicit belief is regular, and we have that

$$\Re_1(\Sigma) \models \mathbf{B}_1 x \quad \text{iff} \quad x \in \text{IMPS}(\Sigma).$$

The theorem then follows from Theorem 1. ■

However else it has been characterized in the past, this theorem establishes that an ATMS can be understood as computing all simplest explanations with respect to this type of implicit belief. Among other things, this guarantees that Poole's account of abduction (with the addition of the notion of simplicity defined here) also specifies the task performed by an ATMS.

5 Case 2: Explicit belief

The second notion of belief we consider is a variant of the one introduced in [Levesque, 1984] called *explicit belief.* We use B_E as the belief operator for beliefs of this type.

In the final paper, we will consider a very different way of generalizing the ATMS to handle arbitrary sentences.

The motivation behind explicit belief was to study a form of belief that was more computationally tractable than implicit belief, but remained defined in terms of truth conditions on the sentences believed. Since a sentence is implicitly believed if it comes out true at each element of a set of assignments (or alternatively, accessible possible worlds), it follows that implicit belief is closed under logical consequence. For explicit belief, instead of using assignments, we use *situations,* which can be taken to be total functions from the *literals* to {0,1}, such that for every *p,* at least one of *p* or *p* is assigned to 1.[12] We can think of assignments as those situations where *s(p)* = 1 — *s(p)* for every letter *p.* But because not every situation is an assignment, we must define truth support recursively over sentences *and* their negations:

$$s \models p \text{ iff } s(p) = 1; \qquad s \models \neg p \text{ iff } s(\overline{p}) = 1;$$
$$s \models (\alpha \wedge \beta) \text{ iff } s \models \alpha \text{ and } s \models \beta;$$
$$s \models \neg(\alpha \wedge \beta) \text{ iff } s \models \neg\alpha \text{ or } s \models \neg\beta;$$
$$s \models \neg\neg\alpha \text{ iff } s \models \alpha.$$

An epistemic state for explicit belief is modeled by a set of situations where we have the following:

$$e \models \mathbf{B_E}\alpha \text{ iff for every } s \in e,\ s \models \alpha.$$

As in [Levesque, 1984], it is also useful to talk about the implicit beliefs of e:

$$e \models \mathbf{B_I}\alpha \text{ iff for every assignment } s \in e,\ s \models \alpha.$$

As before, $R_E(KB)$ is modeled by the set of all situations that satisfy every element of KB. What is explictly believed in such a state is not what logically follows from KB, but rather what is *tautologically entailed* by the KB (once tautologies are taken into account) in the sense of Relevance Logic [Anderson and Belnap, 1975, Dunn, 1976]. More precisely, if $e = \mathfrak{R}_{\mathbf{E}}(KB)$, we have that $e \models \mathbf{B_E}\alpha$ iff KB U T tautologically entails a, where T is the set of all clauses of the form {p,p}.

5.1 Limited abductivc reasoning

To establish what form of abductive reasoning is appropriate for explicit belief, we need something that will play the role that IMPS(E) played for implicit belief:

Definition 10
EXPS(E) = $\{y \mid y \text{ is tautologous or } \exists y^* \in \Sigma,\ y^* \subseteq y\}$.

The abductive reasoning we will use for explicit belief is the same as that performed by the ATMS, but using EXPS(E) instead of IMPS(E):

Definition 11 $\mathbf{abd}[\Sigma, \beta] = \nabla(\mathbf{EXPS}(\Sigma), \mathbf{CNF}(\beta))$.

To see the difference between this procedure and the ATMS, suppose that KB1 = $\{\{q\}, \{\overline{s} \vee p\}, \{\overline{p} \vee \overline{q} \vee r\}\}$. In this case, atms[KBi,r]={r,s,{pA q}}, so there are three simplest explanations for r wrt implicit belief; but abd[KB_1, r]={r, *{p* A *q}},* so *s* is *not* a simplest explanation for r wrt to explicit belief. The difference is that whereas *(s* V r) is implicitly believed (since it follows from KB_1), it is not explicitly believed. In other words, unlike the ATMS, abd[Σ,β] will not chain backwards to see what might explain β, and this is exactly what is required for explicit belief:

Theorem 4 $\text{EXPLAIN}_E[\mathfrak{R}_{\mathbf{E}}(\Sigma), \beta] = \|\mathbf{abd}[\Sigma, \beta]\|$.

Proof: Like implicit belief, explicit belief is regular. Also we have that

$$\mathfrak{R}_{\mathbf{E}}(\Sigma) \models \mathbf{B_E}x \quad \text{iff} \quad x \in \mathbf{EXPS}(\Sigma).$$

The theorem then follows from Theorem 1. ■

This theorem establishes that abd[E, β] correctly calculates all simplest explanations with respect to this type of explicit belief.

But why should we care about a procedure that cannot find some perfectly reasonable explanations that can be found by an ATMS? The problem is that we may have to wait too long for an ATMS to find them. This has caused researchers to look for parallel realizations of the procedure [Dixon and de Kleer, 1988]. But this is not just an ATMS *implementation* problem; the *task* it performs is inherently difficult: in general, there will be an exponential number of clauses to find,[13] and just deciding if {p,p} has *any* explanations at all is equivalent to determining whether or not the set of clauses E is satisfiable. So although (a parallel version of) the ATMS may work fine in many application areas, as a general-purpose mechanism for abductive reasoning, it has serious computational drawbacks.

On the other hand, just as explicit belief is easier than implicit belief when it comes to deductive reasoning, a similar result carries over to abductive reasoning:

Theorem 5 *If* KB *is in CNF, there is an* O(|KB|-|X|) *algorithm for calculating* abd[KB, x].

Proof: We use the fact that

$$\text{abd[KB,z]} = \{(\overline{y} - \overline{x}) \mid y \in \mu\mathbf{EXPS}(\mathbf{KB}),\ x \cap y \neq \emptyset\}.$$

We construct the answer as follows: cycle through the elements of μKB, and for each *y* that is not tautologous and that has an intersection with x, put *(y — x)* into a set *T.* Then, for each *m* € x, put *m* into T, unless *{m}* ∈ KB. Finally, return z for each $z \in \mu$T. ■

So for single clauses anyway, abductive reasoning for explicit belief is considerably easier than abductive reasoning for implicit belief.

For arbitrary sentences , the case is not so clear even if *(3* is in CNF. Although we can quickly calculate abd[E, *x]* for each clause *x* in β, putting the answers together involves converting a sentence into DNF:

Theorem 6 *Suppose* $\text{CNF}(\beta) = \{x_1, \ldots x_n\}$.

$$\text{abd}[\Sigma, \beta] = \mu\{\overline{z} \mid \forall y \in \text{EXPS}(\Sigma),\ y \not\subseteq z \text{ and } \overline{z} \in \text{DNF}(\textstyle\bigwedge_{i=1}^{n} \bigvee \mathbf{abd}[\Sigma, x_i])\}.$$

Proof: The proof will appear in the final paper. ■

To see how this works, let KB_2 be $KB_1 \cup \{\{\overline{a} \vee b\}, \{\overline{c} \vee b\}\}$, where KB_1 is defined above. Then $\mathbf{abd}[KB_2, (r \wedge b)]$ can be computed by calculating $\bigvee\mathbf{abd}[KB_2, r]$ (as above), which gives $(r \vee (p \wedge q))$, then $\bigvee\mathbf{abd}[KB_2, b]$, which gives $(a \vee b \vee c)$,

[12] This restriction on situations was not present in [Levesque, 1984]. It has the effect of making explicit belief similar to the knowledge retrieval of [Frisch, 1988] in that tautologies are always believed. This does not adversely affect the desirable computational properties of explicit belief, since for (non-quantificational) CNF, tautologies can be detected in linear time.

[13] In an unpublished note, David McAllester shows that this remains true even when Σ is a set of Horn clauses.

and then conjoining and putting the result into DNF, which gives

$$\{\{p \wedge q \wedge a\}, \{p \wedge q \wedge b\}, \{p \wedge q \wedge c\}, \{r \wedge a\}, \{r \wedge b\}, \{r \wedge c\}\}.$$

The only potential difficulty here is calculating the DNF. When β has very few clauses, or when almost all of the abd[Σ,x] return fewer than 2 simplest explanations, the entire operation will be fast. But to *guarantee* that it will work well in all cases appears to require an even more restricted form of belief.[14]

6 From explicit to implicit belief

One of the reasons for introducing explicit belief in [Levesque, 1984] was to specify a tractable deductive service for Knowledge Representation in terms of a set of beliefs which, unlike the implicit ones, could always be reliably computed. However, one difficulty with this whole approach is how exactly to go beyond what is explicitly believed. When deliberately trying to solve a problem (in what is called *puzzle mode* in [Levesque, 1988]), it is necessary to combine beliefs and follow through on their consequences in a controlled and systematic way. If all that is available at the knowledge level is a way of finding out if something is explicitly believed and a way of finding out if something is implicitly believed (in one very large unsupervised step), there is nothing the agent can do to begin exploring in a controlled way the implications of what is explicitly believed. For instance, the agent cannot simply perform theorem proving over what is known without access to the sentences at the symbol level used to represent that knowledge.

With a limited abduction operation, on the other hand, there is a way of moving under the control of the agent from the explicit beliefs towards the implicit beliefs. To find out if a sentence is implicitly believed, the procedure (roughly) is this: first find out ifβ is explicitly believed; if it is, then exit with success; otherwise, calculate the full (explicit) explanation for /?; if there is none or it is trivial, then exit with failure; otherwise replace *β* by the explanation, and repeat. In other words, the procedure deals with the following questions, starting with some ft^0*:* according to what is believed,

is ft^0 true? what would it take for ft^0 to be true? (call that β^1)
is ft^1 true? what would it take for β^1 to be true? (call that ft^2)
is ft^2 true? *etc.*

This "backward-chaining" procedure terminates when it either finds something that is believed or fails to find a non-trivial explanation. Each step in this procedure is tractable,[15] and the agent can exit the loop if it seems to be taking too long relative to the importance of the original question. More formally, we have the following:

Definition 12 For any epistemic state e and any β from $\mathcal{L}$, a sequence of sentences β_e^k, $k = 0, 1, 2, \ldots$ is defined by $\beta_e^0 = \beta$ and $\beta_e^{k+1} = \bigvee \text{EXPLAIN}_{\mathbf{E}}[\![e, \beta^k]\!]$.[16]

Theorem 7 $e \models \mathbf{B}_1\beta$ *iff for some* k, $e \models \mathbf{B}_{\mathbf{E}}\beta_e^k$.

Proof: The proof is based on the following:

> **Lemma 7.1** *If* Σ *is satisfiable,* $\Sigma \cap \Gamma = \emptyset$, *and there exists a linear set-of-support resolution refutation of* $\Sigma \cup \Gamma$, *with* Γ *as the set of support, then* $e \models \mathbf{B}_{\mathbf{E}}\alpha^k$ *where* $e = \Re_{\mathbf{E}}(\Sigma)$, $\alpha = \bigvee\{\overline{x} \mid x \in \Gamma\}$, *and* k *is the depth of the refutation tree.*
>
> In the final paper, we prove this lemma and the theorem. ■

So a sentence is implicitly believed iff it is accounted for *ultimately* by something that is explicitly believed.

To see this in action, let $KB_3 = KB2 \cup \{\{s\}, \{a \vee c\}\}$, where KB2 is defined above, and let $e = \Re_{\mathbf{E}}(\mathbf{KB_3})$, the epistemic state represented by KB3. Although $(r \wedge b)$ is not explicitly believed in state e, it is implicitly believed and so should be derivable. First we set *(3°* to $(r \wedge b)$, and compute $\beta^1 = \text{EXPLAIN}_{\mathbf{E}}[\![e, \beta^0]\!] = \mathbf{abd}[KB_3, \beta^0]$, which is

$$\{\{p \wedge q \wedge a\}, \{p \wedge q \wedge b\}, \{p \wedge q \wedge c\}, \{r \wedge a\}, \{r \wedge b\}, \{r \wedge c\}\}.$$

as presented above for KB_2. Putting this into CNF, we get $\{\{a \vee b \vee c\}, \{q \vee r\}, \{p \vee r\}\}$.[17] Notice that the first two clauses of β^1 are already explicitly believed by KB_3. Now calculate $\beta^2 = \text{EXPLAIN}_{\mathbf{E}}[\![e, \beta^1]\!] = \mathbf{abd}[KB_3, (p \vee r)] = \{r, p, s\}$. But then β^2 is explicitly believed (since s is), and so we are done. Notice how the iterative procedure works its way hack to s the way an ATMS would, but now in bite-sized pieces under the control of the agent.[18]

This theorem thus has the following perhaps surprising conclusion: we can determine if something is a logical consequence of what is (explicitly) believed without ever getting access to the set of sentences that are believed. We need only be able to ask for any specific sentence two questions: is it believed? and if not, what would be sufficient to account for it, according to what is believed?

The theorem also provides for the first time a knowledge-level account, that is, an account that is independent of how knowledge is symbolically represented, of how a limited notion of belief can be extended systematically to include all of its logical consequences. It also suggests a knowledge-level account of how an agent's beliefs could be made to evolve deductively over time: starting with some beliefs in some state e0, the agent would believe a in state efc+i iff he believed an explanation of *a*

It appears that a type of belief that is regular except for condition 4, closure under conjunction, does the trick here, but this needs further investigation.

Strictly speaking this is not true because of the DNF problem noted in the previous section. I suspect, however, that the procedure will also work for the more restricted notion of belief, but this has yet to be established.

We are abusing notation here in treating the result of EXPLAIN as a set of sentences.

In practice, one would not want to iterate an abductive procedure that takes the trouble of putting its answer into DNF, since the next step of the iteration requires an argument that is in CNF.

[18] Similar iterative techniques, we suspect, will lead to a procedure for full (implicit) abductive reasoning, as a controlled alternative to the ATMS itself.