Prediction is Deduction
but
Explanation is Abduction

Murray Shanahan

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

December 1988
Revised April 1989

In Proceedings IJCAI 89, pages 1055-1060.

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.

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 Δ, and the causes of Δ are amongst the logical consequences G such that $T \wedge \Delta \vdash G$. 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 Δ and the disease which causes those symptoms is a logical consequence of $T \wedge \Delta$.

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 in 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 $T \wedge \Delta \vdash G$ are both predictions and explanations. That is, G represents projections both forwards and backwards from Δ.

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 Δ by finding the logical consequences G such that $T \wedge \Delta \vdash G$. For explanation, events and properties are represented by G, and the task is to find sets of events Δ which could have caused G, in other words, to find Δ's such that $T \wedge \Delta \vdash G$. 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 t_1. In order to apply default persistence to conclude that it still holds at a later time t_2, 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 I introduce a shortened form of the Event Calculus of Kowalski and Sergot ([9]), which is similar to that presented by Kowalski ([8]). To demonstrate its practical realisability, I describe an abductive mechanism which is related to the techniques of Finger and Genesereth ([5]) and Cox and Pietrzykowski ([2]), and is a simplification of the mechanism described by Eshghi ([3]), tailored for the shortened form of the Event Calculus.

The Event Calculus

In Kowalski and Sergot's Event Calculus ([9]) and its variants (Kowalski [8]), 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 in [9]. 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 *clipped(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*.

initiates(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 act(E,move(X,Y)) and Z≠Y (2.4)

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 *initiates* and *terminates* clauses, or can be expressed as integrity constraints (Eshghi [3]).

The importance of supplying a clear semantics for formulations of default persistence has been demonstrated by Hanks and McDermott ([6]). 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.* [1], Przymusinski [14]). 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 . . . and T1 < T2 and . . . and not holds-at(P',T1) (3.1)

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',t1) (3.2)

Since times are ordered, any set of such clauses is locally stratified (Przymusinski [13]), 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 ([6]).

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-holds-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 matter again.

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 Δ 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 $T \wedge \Delta$. In other words prediction is determining members of G where $T \wedge \Delta \vdash G$.

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 *Δ* which would explain a set of properties *G* expressed as *holds-at* clauses. Then we wish to find *Δ*'s such that $T \wedge \Delta \models G$, and this is abduction.

We can be a bit more precise about what sorts of *Δ* constitute good explanations. First, *Δ* 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 $\Delta^* \subset \Delta$ and $T \wedge \Delta^* \models G$. There can, of course, be many minimal *Δ*'s. A third criterion for a good explanation is that it should postulate the fewest events possible. This suggests a preference relation on *Δ*'s such that *Δ1* is preferable to *Δ2* if it contains fewer *happens* clauses. Of course, there may still be many equally preferable minimal *Δ*'s. Clauses which appear in all *Δ*'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 *Δ* is a set of defeasibly sufficient conditions for *G*. They are only defeasibly sufficient, because of default persistence — the addition of further events to *Δ* could mean that *G* is no longer explicable by *Δ*.

Let me summarise. We have a set of axioms which, using negation-as-failure, embodies a notion of default forwards persistence, and which has a clear semantics. The domain theory is strictly causal and deterministic. What is true in the past fully determines what is true in the future, but not the other way around. Given a course of events, the properties which hold as causal consequences are logical consequences of the domain theory. It would be a conceptual confusion to attempt to add axioms which render what is true in the past a logical consequence of what is true in the future. Furthermore, it would be wasted effort, since all the knowledge that is required for both prediction and explanation is captured within the set of *initiates* and *terminates* clauses and other causal laws which constitutes the domain theory. Finding possible explanations is abduction rather than deduction.

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 t_1. In the absence of any further information, what inferences may we reasonably make about a time t_2 after t_1? The usual notion of

default persistence which licenses the inference that p still holds at t_2, 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 t_1, and then default persistence is applied to conclude that p still holds at t_2. Suppose that the domain theory T comprises Axioms (1.1) to (2.4), that we have a set of axioms Δ 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

$$holds\text{-}at(on(a,x),t_1) \qquad (4.1)$$

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

$$happens(e) \qquad (5.1)$$
$$act(e,move(a,x)) \qquad (5.2)$$
$$e < t_1 \qquad (5.3)$$

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 t_1 and through any time t_2 after t_1. 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 Δ then it would not of course be necessary to add anything to Δ.

Suppose that in addition to (4.1), we are also told that the block a is at location y at time t_3 which is after t_1. So we have

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

$t_1 < t_2 < t_3$ (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 t_1, we assume that it still holds at any later time t_2 unless we have reason to believe that it changes some time between t_1 and t_2. But in this case, we know that at some time between t_1 and t_3 the block ceases to be at location *x* and starts to be at location *y*. In fact, since we do not know when between t_1 and t_3 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 ([7]), and many approaches to default persistence do not deal with it correctly. For example, with Shoham's logic ([16]), 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 t_3.

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 [11], Lifschitz and Rabinov [10]). 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 $T \mathbin{X} \Delta \mathbin{p} G$ 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') (7.1)

act(e',move(a,y)) (7.2)

$e' < t_3$ (7.3)

$t_1 < e'$ (7.4)

With the addition of (7.1) to (7.4) to Δ, because the relative ordering of e' and t_2 is not known, default persistence no longer licenses the conclusion that *holds-at(on(a,x),*t_2*)*. Axioms (7.1) to (7.4) will be present in any Δ which explains *G*, and can be thought of as the necessary conditions for *G* given *T*. A more complicated example might yield many Δ's, and each such Δ 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 t_1. In the absence of any further information, what inferences may we reasonably make about a time t_0 before t_1? The three assumptions which license the default inference that *p* still holds at a time t_2 after t_1 do not apply to a time before t_1. With the additional assumption that properties require explanations, we conclude that some event must have occurred to initiate *p*. But we have no idea when that event occurred —

it may have been before or after t_0. So there is no reason to suppose that p holds at t_0. The correct way to deal with persistence is to ensure that it works forwards only.

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 $\leftarrow G_0$, an SLD refutation of $\leftarrow G_0$ is a sequence of goal clauses $\leftarrow G_0 \ldots \leftarrow G_n$ where $\leftarrow G_n$ is the empty clause and each $\leftarrow G_{i+1}$ is obtained from $\leftarrow G_i$ 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 $\leftarrow G_i$ whose selected literal g will not resolve with any clause in T. Usually this means that sequences beginning with $\leftarrow G_0 \ldots \leftarrow G_i$ are not worth exploring. But if we are searching for a set of unit clauses Δ such that $T \cup \Delta \vdash G_0$, then clearly by letting Δ include a unit clause which resolves with g, we can continue the search with $\leftarrow G_{i+1}$ equal to $\leftarrow G_i$ 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 $T \cup \Delta_n \vdash G_0$ and Δ_n mentions only abducibles, a refutation of the form $\leftarrow G_0,\Delta_0 \ldots \leftarrow G_n,\Delta_n$ is constructed, where each $\leftarrow G_i$ is a goal clause, each Δ_i is a set of unit clauses mentioning only abducibles, $\leftarrow G_n$ is the empty clause, Δ_0 is the empty set, and each $\leftarrow G_{i+1},\Delta_{i+1}$ is obtained from $\leftarrow G_i,\Delta_i$ as follows. If g, the selected literal of $\leftarrow G_i$, can be resolved with one of the clauses in T, then a single resolution step is taken as described above and Δ_{i+1} is Δ_i. If g is abducible and cannot be resolved with any clause in T, then G_{i+1} is G_i minus g and Δ_{i+1} is Δ_i plus the unit clause $g' \leftarrow$ where g' is g with all its variables replaced by skolem constants (Cox and Pietrzykowski [2]). If g were not skolemised, all the variables in $g' \leftarrow$ 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 ([4]) and Poole ([12]) 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 [14], Shanahan [15]). 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 $g \leftarrow not\ h'$ 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(E,A)* and *T1 < T2* is abducible. The initial goal clause is of the form $\leftarrow holds\text{-}at(P_1,T_1), \ldots, holds\text{-}at(P_n,T_n)$, 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 $\leftarrow act(E,move(a,b)), \ldots$, 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 $\leftarrow act(e,move(a,b))$. 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.

The above treatment of skolem constants creates another problem. Suppose that the abductive mechanism encounters the goal $x \neq l$ where *l* is a skolem constant. Since *l* can later be replaced by another constant, it could be replaced by *x*, making the goal false. To cope with this, inequalities involving skolem constants have to be treated in a similar way to persistence

assumptions and other negated literals. In effect, they are made defeasible, by recording them and rechecking them whenever skolem constants are replaced by other constants. This corresponds to reading $X \neq Y$ as *not* $X=Y$ where *not* is interpreted as negation-as-failure.

Let us consider a trivial example of this mechanism applied 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),t0)*. The search space for this example is shown in Figure 1. Abduction generates the residue $\Delta'=\{happens(e1),\ act(e1,move(a,x)),\ e1<t0\}$.

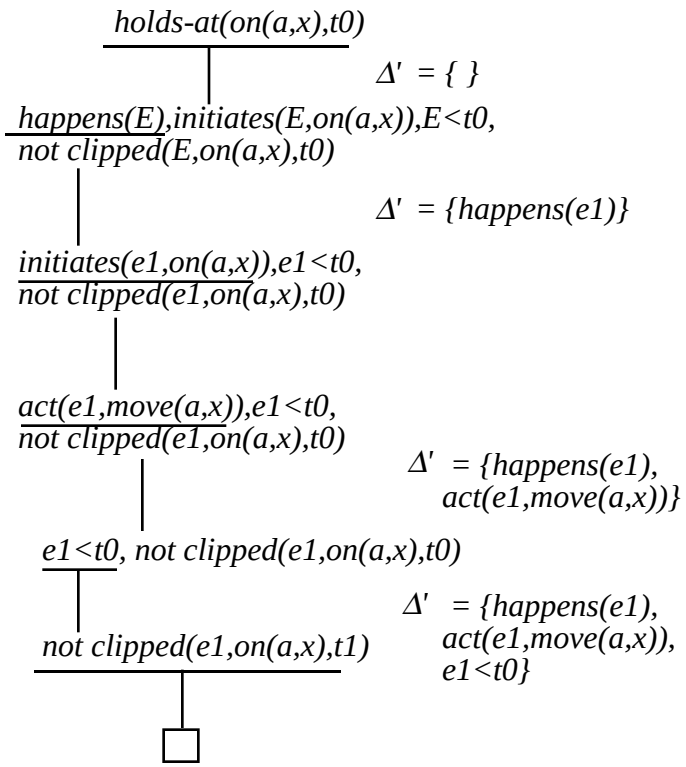

Figure 1.

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 *t1<t2*, and we want an explanation for *holds-at(on(a,x),t0) and holds-at(on(a,x),t2) and holds-at(clear(x),t1)*. 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 $\Delta = \Delta' \times \Delta''$.

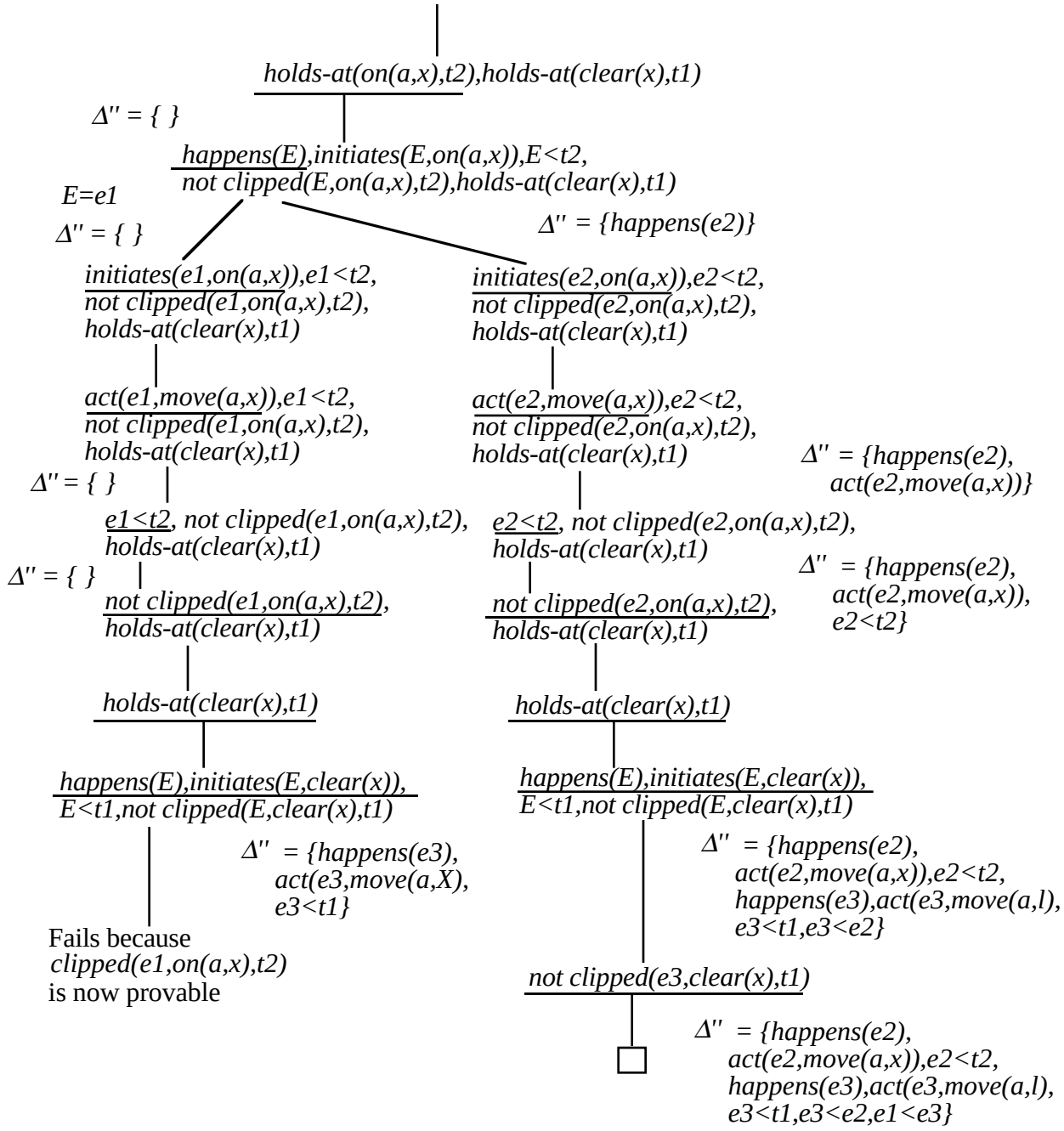

Figure 2.

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 *e1*, or of postulating a new event *e2*. This gives rise to two branches in the search space. Furthermore, *e1* does initiate the property *on(a,x)* and does occur before *t2*, and it cannot yet be shown that *clipped(e1,on(a,x),t2)*. But when, in order to explain *holds-at(clear(x),t1)*, 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(e1,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 *Δ={happens(e1), act(e1,move(a,x)), e1<t0, happens(e2), act(e2,move(a,x)), e2<t2, happens(e3), act(e3,move(a,l)), e3<t1, e3<e2, e1<e3}*. The skolem constant *l* 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),t1)* were not included then the first branch of the search space would succeed with the simplest explanation, postulating only the event *e1* 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 *e1<e2,* it would have been possible to prove *clipped(e3,clear(x),t1)*.

Concluding Remarks

Finger and Genesereth ([5]) 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 ([2]) also describe a related technique. Eshghi ([3]) 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 ([4]), however, present an abduction semantics for negation-as-failure itself, and Poole ([12]) 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.

Morgenstern and Stein ([11]) and Lifschitz and Rabinov ([10]) 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. Some of this work was carried out while working on the Esprit project EQUATOR.

References

[1] 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.

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

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

[4] 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).

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

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

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

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

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

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

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

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

[13] 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.

[14] 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.

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

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