Actions and Events in Interval Temporal Logic*

James F. Allen George Ferguson

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 521

July 1994

Abstract

We present a representation of events and action based on interval temporal logic that is significantly more expressive and more natural than most previous AI approaches. The representation is motivated by work in natural language semantics and discourse, temporal logic, and AI planning and plan recognition. The formal basis of the representation is presented in detail, from the axiomatization of time periods to the relationship between actions and events and their effects. The power of the representation is illustrated by applying it to the axiomatization and solution of several standard problems from the AI literature on action and change. An approach to the frame problem based on explanation closure is shown to be both powerful and natural when combined with our representational framework. We also discuss features of the logic that are beyond the scope of many traditional representations, and describe our approach to difficult problems such as external events and simultaneous actions.

This material is based upon work supported by the National Science Foundation under Grant number IRI-9003841. The Government has certain rights in this material. This work is also supported by ONR/ARPA research grant no. N00014-92-J-1512 and Air Force – Rome Air Development Center research contract no. F30602-91-C-0010.

*This work is to appear in the *Journal of Logic and Computation*, Special Issue on Actions and Processes, 1994.

1 Introduction

Representing and reasoning about the dynamic aspects of the world—primarily about actions and events—is a problem of interest to many different disciplines. In AI, we are interested in such problems for a number of reasons, in particular to model the reasoning of intelligent agents as they plan to act in the world, and to reason about causal effects in the world. More specifically, a general representation of actions and events has to support the following somewhat overlapping tasks:

- Prediction: Given a description of a scenario, including actions and events, what will (or is most likely to) happen?
- Planning: Given an initial description of the world and a desired goal, find a course of action that will (or is most likely to) achieve that goal.
- Explanation: Given a set of observations about the world, find the best explanation of the data. When the observations are another agent's actions and the explanation desired is the agent's plan, and the problem is called plan recognition.

Our claim in this paper is that in order to adequately represent actions and events, one needs an explicit temporal logic, and that approaches with weaker temporal models, such as state spaces (*e.g.*, STRIPS-based approaches) and the situation calculus, either cannot handle the problems or require such dramatic extensions that one in effect has grafted an explicit temporal logic onto the earlier formalism. Furthermore, if one of these formalisms is extended in this way, the temporal logic part will dominate and the original formalism plays little role in the solution. We will primarily defend this position by proposing a specific temporal representation and showing that it can handle a wide range of situations that are often problematic for other formalisms. In particular, here are some of properties of actions and events that we feel are essential to any general representation:

1. Actions and events take time. While some events may be instantaneous, most occur over an interval of time. During this time, they may have a rich structure. For instance, the event of driving my car to work involves a wide range of different actions, states of the world and other complications, yet the activity over that stretch of time can be nicely described as a single event. Because events are extended in time, different events and actions may overlap in time and interact. Thus, if while I am driving to work, a rock pierces the gas tank and the gasoline drains out, I may end up being stranded on the highway rather than arriving at work on time. We should be able to represent and reason about such complex interactions.
2. The relationship between actions and events and their effects is complex. Some effects become true at the end of the event and remain true for some time after the event. For example, when I put a book on the table, this has the effect that the book is on the table for at least a short time after the action is completed. Other effects only hold while the event is in progress. For instance, consider a flashlight with a button for flashing it. The light is on only when the button is being pressed down. Finally,

other effects might start to hold sometime after the event has started and stop holding before it finishes. For example, if I am walking to school along my usual route, at some time during the walk I will be on the bridge crossing the river. This is an effect of the action even though it is not true at the end of it.

3. Actions and events may interact in complex ways when they overlap or occur simultaneously. In some cases, they will interfere with certain effects that would arise if the events were done in isolation. In other cases, the effect of performing the two actions may be completely different that the effects of each in isolation. As a radical case of this, consider a wristwatch that is controlled by two buttons A, and B. Pressing A changes the mode of the display, while B shows the alarm time setting. Pressing A and B simultaneously, however, turns the alarm on or off. The effect of performing the actions simultaneously has no causal relation to the effects of the actions performed in isolation.

4. External changes in the world may occur no matter what actions an agent plans to do, and may interact with the planned actions. Possible external events should be an important factor when reasoning about what effects an action might have. Certain goals can only be accomplished by depending on external events. For example, to sail across a lake, I can put up the sails but I depend on the wind blowing to bring about the event.

5. Knowledge of the world is necessarily incomplete and unpredictable in detail, thus prediction can only be done on the basis of certain assumptions. Virtually no plan is foolproof, and it is important that a formalism makes the necessary assumptions explicit so that they can be considered in evaluating plans.

Our aim is to develop a general representation of actions and events that supports a wide range of reasoning tasks, including planning, explanation, and prediction, but also natural language understanding, and commonsense reasoning in general. Most of the previous work on these problems tends to address only a subset of these problems. The STRIPS-based planners (*e.g.*, TWEAK [Chapman, 1987], SIPE [Wilkins, 1988], SNLP [McAllester and Rosenblitt, 1991]), for instance, only address the planning problem, while work in the situation calculus (*e.g.*, [McCarthy and Hayes, 1969; Baker, 1991]) has primarily focussed on the prediction problem or on using it as an abstract theory for planning (*e.g.*, [Green, 1969]). Natural language work, on the other hand, typically only deals with commonsense entailments from statements about actions and events, sometimes with a focus on plan recognition and explanation (*e.g.*, [Allen, 1994; Hobbs *et al.*, 1993; Schubert and Hwang, 1989]). Our representation is intended to serve as a uniform representation to support all these tasks. As a result, we try to avoid introducing any specific syntactic constructs that support only one reasoning task. Knowledge should be encoded in a way so that it is usable no matter what reasoning task is currently being performed.

Section 2 outlines our intuitions about actions and events, and briefly explores the two predominant models of action: the situation calculus and the state-based STRIPS-style representations. As typically formulated, neither is powerful enough for the issues described above. Section 3 then introduces Interval Temporal Logic, first defining the temporal structure, then introducing properties, events, and actions. Section 4 demonstrates how the interval logic can be used to solve a set of simple problems from the literature in order to facilitate comparison with other approaches. The key place where other approaches have difficulty is in representing external events and simultaneous actions. Section 5 explores the implications of external events in detail, and Section 6 explores interacting simultaneous actions.

2 Representing Actions and Events

Before starting the formal development, we will attempt to describe the intuitions motivating the representation. We will then consider why the most commonly accepted representations of action in AI will not meet our needs.

2.1 Intuitions about Actions and Events

The first issue concerns what an event is. We take the position that events are primarily linguistic or cognitive in nature. That is, the world does not really contain events. Rather, events are the way by which agents classify certain useful and relevant patterns of change. As such, there are very few restrictions on what an event might consist of except that it must involve at least one object over some stretch of time, or involve at least one change of state. Thus the very same circumstances in the world might be described by any number of events. For instance, consider a circumstance in which a ball rolled off a table and bounced on the floor. This already is one description of what actually happened. The very same set of circumstances could also be described as the event in which the ball fell off the table, or in which the ball hit the ground, or in which the ball dropped. Each of these descriptions is a different way of describing the circumstances, and each is packaged as a description of an event that occurred. No one description is more correct than the other, although some may be more informative for certain circumstances, in that they help predict some required information, or suggest a way of reacting to the circumstances.

Of course, the "states" of the world referred to above are also ways of classifying the world, and are not inherent in the world itself either. Thus, the same set of circumstances described above might be partially described in terms of the ball being red. Given this, what can one say about the differences between events and states? Intuitively, one describes change and the other describes aspects that do not change. In language, we say that events occur, and that states hold. But it is easy to blur these distinctions. Thus, while the balling falling from the table to the floor clearly describes change and hence describes an event, what about the circumstance where an agent John holds the door shut for a couple of minutes. While the door remains shut during this time and thus doesn't change state, it seems that John holding the door shut is something that occured and thus is like an event. These issues have been studied extensively in work on the semantics of natural language sentences. While there are many proposals, everyone agrees on a few basic distinctions (*e.g.*, [Vendler, 1967; Mourelatos, 1978; Dowty, 1986]). Of prime relevance to us here are sentences that describe *states* of the world, as in "The ball is red," or "John believes that the world is flat," and sentences that describe general ongoing *activities* such as "John ran for an hour," and *events* such as "John climbed the mountain." Each of these types of sentences has different properties, but the most important distinctions occur in their relation to time. All these sentences can be true over an interval of time. But when a state holds over an interval of time t, one can conclude that the state also held over subintervals of t. Thus, if the ball is red during the entire day, then it is also red during the morning. This property is termed *homogeneity* in the temporal logic literature. Events, on the other hand, generally have the opposite property and are anti-homogeneous: If an event occurs over an interval t, then it doesn't occur over a subinterval of t, as it would not yet be completed. Thus, if the ball dropped from the table to the floor over time t, then over a subinterval of t is would just be somewhere in the air between the table and floor. Activities, on the other hand, fall in between. They may be homogenous, as in the holding the door shut example above, but they describe some dynamic aspect of the world like events. This type of distinction must appreciated by any general purpose knowledge representation for action and change.

The other general point to make is that intervals of time play an essential role in any representation of events. Events are defined to occur over intervals of time, and cannot be reduced to some set of properties holding at instantaneous points in time. Of course, semantically, one can define time intervals as an ordered pair of time points, but the truth conditions must be defined in terms of these ordered pairs, and not in terms of the individual points. Specifically, if an interval were simply a set of points between two points, and truth over an interval was defined in terms of truth over all the points in the interval, then every predicate would have to be homogeneous.

Finally, a word on actions. The word "action" is used in many different senses by many different people. For us, an action refers to something that a person or robot might do. It is a way of classifying the different sorts of things than an agent can do to affect the world, thus it more resembles a sensory-motor program than an event. By performing an action, an agent causes an event to occur, which in turn may cause other desired events to also occur. For instance, I have an action of walking that I may perform in the hope of causing the event of walking to my car. Some theories refer to the event that was caused as the action, but this is not what we intend here. Rather, we will draw on an analogy with the robot situation, and view actions as programs. Thus, performing an action will be described in terms of running a program.

2.2 The Situation Calculus

The situation calculus means different things to different researchers. In its original formulation [McCarthy and Hayes, 1969], which we will call the general theory of the situation calculus, situations are introduced into the ontology as a complete snapshot of the universe at some instant in time. In effect, the situation calculus is a point-based temporal logic with a branching time model. In its most common use, which we will call the constructive situation calculus, it is used in a highly restricted form first proposed by Green [1969], in which the only way situations are introduced is by constructing them by action composition from an initial state. This practice has attracted the most attention precisely because the formalism is constructive—specifically, it can be used for planning. To contruct a plan for a goal G, prove that there exists a situation s in which G holds. In proving this, the situation is constructed by action composition, and thus the desired sequence of actions (the plan) can be extracted from the proof. As others have pointed out (*e.g.*, [Schubert, 1990]), most of the criticisms about the expressibility of the situation calculus concern the constructive form of it rather than the general theory. Our position is that the constructive situation calculus is a limited representation, especially in dealing with temporally complex actions and external events. The general theory, on the other hand, is much richer and can be extended to a model much closer to what we are proposing, but at the loss of its constructive aspect.

To see some of the difficulties, first consider a very simple example. In a domain that reasons about transportation of cargo, we might want to define an action of a train moving some cargo between two cities. Some of the information needed to reason about this action is the following:

1. The train initially starts at the originating city S.
2. The trip typically takes between 4 and 6 hours.
3. The cars must remain attached to the train during the entire trip.
4. The train will be on track segment $A1$ then it will cross junction $J1$, and be on track segment $A2$ for the rest of the trip.
5. The train will end up at the destination city D.

This is all very mundane information, but each fact might be crucial for some planning task. For instance, knowledge of the track segments that the train is on during the trip might be used to avoid having multiple trains on the same track at the same time.

In its constructive form, the situation calculus can only adequately represent properties (1) and (5). Actions are represented syntactically as functions from one situation to the resulting situation, and there appears to be no mechanism for representing information about the duration of actions (as needed for property 2), for representing effects that do not start at the end of the action (property 4), or for representing preconditions that must hold beyond the start time of the action (property 3). In addition, this example didn't include the more difficult problems of representing external events, or interacting simultaneous actions. While there is now a considerable literature on extensions to the formalism to handle some of these problems, most lose the constructivity property and thus become less suitable for practical reasoning tasks such as planning. In addition, most of the extensions involve syntactic extensions to the language that are defined semantically by particular minimization strategies specifically tailored to the problem. As a result, trying to combine solutions to all the problems into a single uniform formalism is remarkably complex, even when remaining focused on carefully hand-tailored sample problems. The chances of integrating such representations into a more general knowledge representation system for tasks such as real-world planning or natural language understanding seem remote.

2.3 The STRIPS Representation

The STRIPS representation [Fikes and Nilsson, 1971] adds additional constraints to the situation calculus model and is used by most implemented planning systems built to date.

```
STACK(a,b)
        preconds:  (Clear a) (Clear b)
        delete:    (Clear b)
        add:       (On a b)
```

Figure 1: Actions as state change in STRIPS

In STRIPS, a state is represented as a finite set of formulas, and the effects of an action are specified by two sets of formulas: the delete list specifies what propositions to remove from the initial state, and the add list specifies the new propositions to add. Together, they completely define the transition between the initial state and the resulting state. Figure 1 shows a simple Blocks World action that involves placing one block on top of another (the `STACK` action). The preconditions on an action indicate when the action is applicable—in this case it is applicable whenever both blocks are clear. The effects state that one block is now on top of the other (the add list) and that the bottom block is not clear (the delete list). The operation for constructing a resulting state applies the delete list first and then asserts the add list.

Like the situation calculus, STRIPS-style actions are effectively instantaneous and there is no provision for asserting what is true while an action is in execution. Also, since the state descriptions do not include information about action occurrences, such systems cannot represent the situation where one action occurs while some other event or action is occurring. The validity of the method comes from the STRIPS assumptions, which assume that the world only changes as the result of a single action by the agent, and that the action definitions completely characterize all change when an action is done. Of course, these assumptions would not be valid if simultaneous actions or external events were allowed.

While discussing the world representation in such models, it is important to include explicitly the nonlinear planners (*e.g.*, [Sacerdoti, 1975; Tate, 1977; Vere, 1983; Chapman, 1987; Wilkins, 1988; McAllester and Rosenblitt, 1991]) as the representational power of these systems is often misunderstood. In particular, the underlying world model used by such systems is essentially the same state-based model as used in STRIPS. For example, Figure 2 shows a very simple nonlinear plan. It represents a partial ordering of actions, where action A must precede B and C, and B and C must precede action D, but actions B and C are unordered with respect to each other. This is actually a compact representation of two distinct linear plans, namely, A, B, C, D in order, or A, C, B, D in order. It does not include the possibility that actions B and C are simultaneous. Nonlinearity is a property of the search process in constructing plans, and not a property of the representation of the world.

Nonlinear planning was developed so that decisions about action ordering could be delayed as long as possible, avoiding the need for backtracking in cases where it was not necessary. But information about a particular state can only be obtained by inferring what remains true throughout all the possible linear orderings. Thus, if we look at the example in Figure 2, the only assertions the system could make about the state after D would be those assertions that were true both in the state obtained by applying A, B, C, and D in

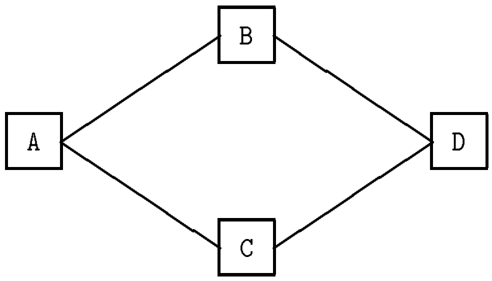

Figure 2: A simple nonlinear plan

sequence *and* in the state obtained by applying A, C, B, and D in sequence. For example, if B were "paint the room blue," and C were "paint the room crimson," then the color of the room after D would be undetermined, since the final color of the room given order A, B, C, D would be crimson, whereas the color of the room given the order A, C, B, D would be blue. If, instead, B were "paint the chair blue," then the result of the nonlinear plan would be that the room is crimson (from C) and the chair is blue (from B), since this is the resulting state given either the ordering A, B, C, D or the ordering A, C, B, D.

This final example suggests an extension to handle some cases of simultaneous action. In particular, in the last example above, the effect of performing the two actions B and C simultaneously could be the same as the result computed for any of the action orderings allowed by the nonlinear plan. Tate [1977], Vere [1983], and Wilkins [1988] have all used this extension in their systems and allow simultaneous action when the two actions are independent of each other. In such cases, the effect of the two acts performed together is the simple union of the individual effects of the acts done in isolation. Wilkins develops an interesting approach using the notion of resources to reason about the independence of actions. The problem with this solution is that it excludes common situations of interest in realistic domains, namely where simultaneous actions are not independent, such as when they have additional synergistic effects or interfere with each other.

The STRIPS assumptions appear to be fundamentally incompatible with interacting simultaneous actions and external events. They hold only in worlds with a single agent, who can only do one thing at a time, and where the world only changes as the result of the agent's actions. However, they are so ingrained in the planning literature that many researchers don't even acknowledge their limitations. In some sense, they have become part of the definition of the classical planning problem.

3 Interval Temporal Logic

Having described our motivations, we now start the development of the temporal logic. We start by describing the basic temporal structure to be used in the logic, namely the interval representation of time developed by Allen [1983; 1984] and discussed in detail in [Allen and Hayes, 1989]. We then describe the temporal logic used to represent knowledge of properties,

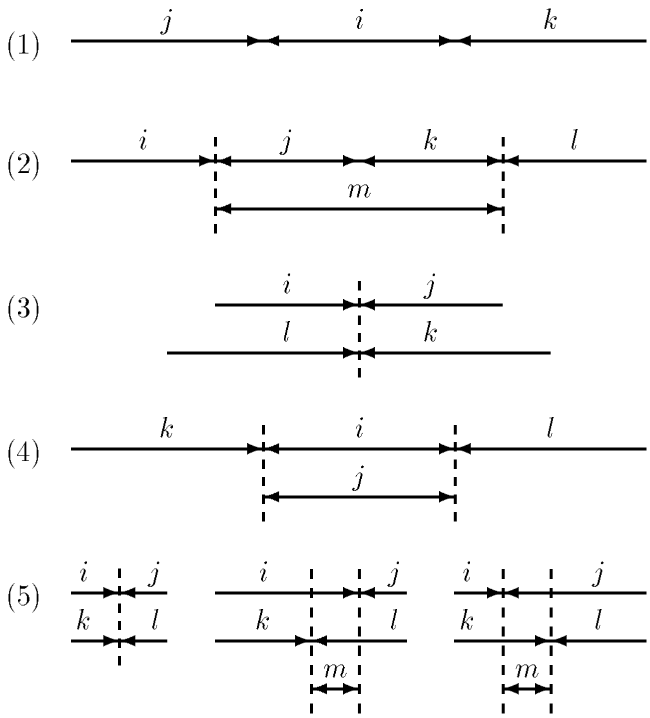

Figure 3: The axiomatization of time periods

events, and actions. We conclude this section with a comparison of related formalisms. Subsequent sections will explore how the representation supports reasoning about events and actions, especially in complex situations with external events and simultaneous actions.

3.1 The Structure of Time

The temporal structure we assume is a simple linear model of time. Notions of possibility that are introduced in branching time models or the situation calculus would be handled by introducing a separate modal operator to represent possibility explicitly. Since such a modal operator is needed in general anyway, there seems no need to build it into the temporal structure. The temporal theory starts with one primitive object, the time period, and one primitive relation: $Meets$.

A time period intuitively is the time associated with some event occuring or some property holding in the world. Intuitively, two periods m and n meet if and only if m precedes n, yet there is no time between m and n, and m and n do not overlap. The axiomatization of the $Meets$ relation is as follows, where i, j, k, l, and m are logical variables restricted to time periods. The axioms are presented graphically in Figure 3. First, there is no beginning or ending of time and there are no semi-infinite or infinite periods. In other words, every period has a period that meets it and another that it meets:

$$\forall i \,.\, \exists j, k \,.\, Meets(j,i) \wedge Meets(i,k). \tag{1}$$

Second, periods can compose to produce a larger period. In particular, for any two periods that meet, there is another period that is the "concatenation" of them. This can be axiomatized as follows:

$$\forall i, j, k, l \,.\, Meets(i, j) \land Meets(j, k) \land Meets(k, l) \supset \exists m \,.\, Meets(i, m) \land Meets(m, l). \tag{2}$$

As a convenient notation, we will often write $j + k$ to denote the interval that is the concatenation of intervals j and k. This functional notation is justified because we can prove that the result of $j + k$ is unique [Allen and Hayes, 1989].

Next, periods uniquely define an equivalence class of periods that meet them. In particular, if i meets j and i meets k, then any period l that meets j must also meet k:

$$\forall i, j, k, l \,.\, Meets(i, j) \land Meets(i, k) \land Meets(l, j) \supset Meets(l, k). \tag{3}$$

These equivalence classes also uniquely define the periods. In particular, if two periods both meet the same period, and another period meets both of them, then the periods are equal:

$$\forall i, j, k, l \,.\, Meets(k, i) \land Meets(k, j) \land Meets(i, l) \land Meets(j, l) \supset i = j. \tag{4}$$

Finally, we need an ordering axiom. Intuitively, this axiom asserts that for any two pairs of periods, such that i meets j and k meets l, then either they both meet at the same "place," or the place where i meets j precedes the place where k meets l, or vice versa. In terms of the meets relation, this can be axiomatized as follows, where the symbol "$\otimes$" means "exclusive-or":

$$\begin{aligned} &\forall i, j, k, l \,.\, (Meets(i, j) \land Meets(k, l)) \supset \\ &\quad Meets(i, l) \otimes (\exists m \,.\, Meets(k, m) \land Meets(m, j)) \otimes \\ &\qquad (\exists m \,.\, Meets(i, m) \lor Meets(m, l)). \end{aligned} \tag{5}$$

Many of the properties that are intuitively desired but have not yet been mentioned are actually theorems of this axiomatization. In particular, it can be proven that no period can meet itself, and that if one period i meets another j then j cannot also meet i (*i.e.*, finite circular models of time are not possible).

With this system, one can define the complete range of the intuitive relationships that could hold between time periods. For example, one period is before another if there exists another period that spans the time between them, for instance:

$$Before(i, j) \equiv \exists m \,.\, Meets(i, m) \land Meets(m, j).$$

Figure 4 shows each of these relationships graphically (equality is not shown). We will use the following symbols to stand for commonly-used relations and disjunctions of relations:

$Meets(i, j)$	$i : j$		
$Before(i, j)$	$i \prec j$	$Before(i, j) \lor Meets(i, j)$	$i \prec\!: j$
$During(i, j)$	$i \sqsubset j$	$During(i, j) \lor i = j$	$i \sqsubseteq j$

Relation	Inverse
$Before(i,j)$	$After(j,i)$
$Meets(i,j)$	$MetBy(j,i)$
$Overlaps(i,j)$	$OverlappedBy(j,i)$
$Starts(i,j)$	$StartedBy(j,i)$
$During(i,j)$	$Contains(j,i)$
$Finishes(i,j)$	$FinishedBy(j,i)$

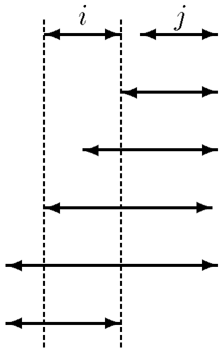

Figure 4: The possible relations between time periods (equality not shown)

Finally, an important relationship between two periods is *Disjoint*: two intervals are disjoint if they do not overlap in any way. We write this as "$i \bowtie j$" and define it by

$$i \bowtie j \equiv i \prec: j \vee j \prec: i.$$

The computational properties of the interval calculus and algorithms for maintaining networks of temporal constraints are presented in [Allen, 1983; Vilain and Kautz, 1986; Vilain *et al.*, 1990].

A period can be classified by the relationships that it can have with other periods. For example, we call a period that has no subperiods (*i.e.*, no period is contained in it or overlaps it) a *moment* and a period that has subperiods, an *interval.* In addition, we can define a notion of time point by a construction that defines the beginning and ending of periods. It is important to note that moments and points are distinct and cannot be collapsed. In particular, moments are true periods and may meet other periods, and if $Meets(i,m) \wedge Meets(m,j)$ for any moment m, then i is before j. Points, on the other hand, are not periods and cannot meet periods. Full details can be found in [Allen and Hayes, 1989].

Any semantic model that allows these distinctions would be a possible model of time. In particular, a discrete time model can be given for this logic, where periods map to pairs of integers $\langle I, J\rangle$ where $I < J$. Moments correspond to pairs of the form $\langle I, I+1\rangle$, and points correspond to the integers themselves. A similar model built out of pairs of real numbers does not allow moments. A more complex model can be specified out of the reals, however, that does allow continuous time, or models that are sometimes discrete and sometimes continuous are possible. Ladkin and Maddux [Ladkin and Maddux, 1988] have characterized the set of possible models as precisely the arbitrary unbounded linear orders.

3.2 Interval Temporal Logic

The most obvious way to add times into a logic is to add an extra argument to each predicate. For example, in a nontemporal logic, the predicate *Green* might denote the set of green objects. Thus, a formula such as $Green(frog13)$ would be true only if the object named by the term, $frog13$, is in the set of green objects. To make this a temporal predicate, a time argument is added and $Green$ now denotes a set of tuples consisting of all green objects with the times at which they were green. Thus, the proposition $Green(frog13, t1)$ is true only if the object named by $frog13$ was green over the time named by $t1$.

By allowing time intervals as arguments, we open the possibility that a proposition involving some predicate P might be neither true nor false over some interval t. In particular, consider a predicate P that is true during some subinterval of t and also false in some other subinterval of t. In this case, there are two ways we might interpret the negative proposition $\sim P(t)$. In the weak interpretation, $\sim P(t)$ is true if and only if it is not the case that P is true throughout interval t, and thus $\sim P(t)$ is true if P changes truth-values during t. In the strong interpretation of negation, $\sim P(t)$ is true if and only if P is false throughout t, and so neither $P(t)$ nor $\sim P(t)$ would be true in the above situation. Thus, a logic with only strong negation has truth gaps.

We take the weak interpretation of negation as the basic construct, as do Shoham [1987] and Bacchus *et. al.* [1989], to preserve a simple two-valued logic. Weak negation also seems to be the appropriate interpretation for the standard definition of implication. In particular, the formula $P(t) \supset Q(t')$ is typically defined as $\sim P(t) \vee Q(t')$. Since we want the implication to mean "whenever P is true over t, then Q is true over t'," this is best captured by the weak negation form of the equivalent formula. With weak negation, $\sim P(t) \vee Q(t')$ says that either P is not true throughout t (but might be true in some subintervals of t), or Q is true over t'. This seems the right interpretation. Of course, we can still make assertions equivalent to the strong negation. The fact that P is false throughout t can be expressed as

$$\forall t' \,.\, t' \sqsubseteq t \supset \sim P(t').$$

This is a common enough expression that we will introduce an abbreviation for the formula, namely $\neg P(t)$ (that is, the symbol "$\neg$" means strong negation). This way we obtain the notational convenience of strong negation while retaining the simpler semantics of a logic with no truth gaps.

There are several characteristics of propositions that allow them to be broadly classified based on their inferential properties. These distinctions were originally proposed by Vendler [1967], and variants have been proposed under various names throughout linguistics, philosophy, and artificial intelligence ever since (*e.g.*, [Mourelatos, 1978; Allen, 1984; Dowty, 1986; Shoham, 1988]). For the most part we will not be concerned with all the distinctions considered by these authors. However one important property mentioned previously is homogeneity. Recall that a proposition is homogeneous if and only if when it holds over a time period t, it also holds over any period within t. In the current formulation, this property is defined by a family of axiom schemata, one for each arity of predicate. For all homogeneous predicates P of arity $n+1$:

Homogeneity Axiom Schema

$$\forall x_i, t, t' \,.\, P(x_1, \ldots, x_n, t) \wedge t' \sqsubseteq t \supset P(x_1, \ldots, x_n, t').$$

All predicates will be homogeneous in what follows except when explicitly noted. The following useful theorem follows from the definition of strong negation and the homogeneity property, for any homogeneous predicate P:

DISJ $\forall t, t' . P(t) \wedge \neg P(t') \supset t \bowtie t'$.

That is, two intervals over which a predicate has different truth values (with respect to strong negation) must be disjoint.

The use of weak negation as the basic construct might also allow intervals over which $\char`~P$ holds but for no subinterval of which does $\neg P$ hold. Within such an interval, there would be an infinite telescoping sequence of intervals within which P alternately holds and does not hold. We rule out such intervals with the following axiom (schema) of discrete variation:

Discrete Variation Axiom Schema

$$\forall t . \char`~P(t) \supset \exists t' . t' \sqsubseteq t \wedge \neg P(t').$$

Hamblin [1972] refers to this axiom and the homogeneity axiom as characterizing the "phenomenal" predicates. Further discussion of discrete variation and its relation to infinite series in calculus and physics can be found in [Davis, 1992].

Finally, combining the axiom of discrete variation with the homogeneity axioms allows us to derive the following theorem useful for reasoning about when properties change truth values:

TRPT $\forall t, t' . P(t) \wedge \neg P(t') \wedge t \prec: t' \supset$
$\exists T, T' . P(T) \wedge \neg P(T') \wedge T : T' \wedge t' \sqsubseteq T'$

Defining a *transition point* to be the existence of adjacent intervals for which a property has complementary truth values, this axiom states that if a property has changed truth value, then there must be transition point where it does so.

3.3 The Logic of Events

The logic developed thus far is still insufficient to conveniently capture many of the circumstances that we need to reason about. In particular, we need to introduce events as objects into the logic. There are many reasons for this, and the most important of these are discussed in the remainder of this section.

Davidson [1967] argued that there are potentially unbounded qualifications that could be included in an event description. For example, the event of Jack lifting a particular ball might be asserted to occur at some time by a predicate LIFT, as $\textsc{Lift}(jack34, ball26, t1)$. The problem now arises in representing the event "Jack lifted the ball onto the table." Either we need to add another argument to the LIFT predicate, or we need to introduce a new predicate that represents a variant of lifting that includes an extra argument. Neither is satisfactory. In the former case, all predicates describing event occurrences will contain a large number of argument positions, typically unspecified in any particular event description and thus requiring a large number of existential variables. In the latter case, we have a large class of predicates all asserting essentially the same thing. If we could put a bound on the number of argument positions needed, then one of these solutions might be viable. But in fact, it seems we could always add additional information about the event that has not yet been captured by a parameter, forcing us to add another. In natural language, this creates particular difficulty when representing adverbial modifiers. Davidson suggested the solution of reifying events, whereby additional modifiers would simply become additional predications on the event. Thus, the event of Jack lifting the ball onto the table with the tongs might be represented as

$$\exists e \,.\, \textsc{Lift}(jack34, ball26, t1, e) \wedge dest(e) = table5 \wedge instrument(e) = tongs1.$$

The issue of reifying events is not only an issue for representing natural language meaning, however. A sophisticated plan reasoning system also needs to represent and reason about events of similar complexity. In addition, in many forms of plan reasoning, the system must be able to distinguish events even though it does not have any information to distinguish them. For instance, in a plan recognition task, an agent might know that two separate lifting events occurred, but it knows little else. In a reified logic this is easy to state, namely, as

$$\exists e_1, e_2 \,.\, \textsc{Lift}(e_1) \wedge \textsc{Lift}(e_2) \wedge e_1 \neq e_2.$$

However, in a nonreified logic such a situation would have to be represented by a complex formula growing in size with the number of arguments. Thus, if the $\textsc{Lift}$ predicate took five arguments, we would need a formula such as

$$\exists t_1, t_2 \,.\, \forall a_1, \ldots, a_8 \,.\, \textsc{Lift}(a_1, a_2, a_3, a_4, t_1) \wedge \textsc{Lift}(a_5, a_6, a_7, a_8, t_2) \wedge \\ (a_1 \neq a_5 \vee a_2 \neq a_6 \vee a_3 \neq a_7 \vee a_4 \neq a_8 \vee t_1 \neq t_2).$$

Besides being cumbersome, this formulation is committed to always being able to distinguish lifting events on the basis of the five arguments given. If Davidson's argument is accepted and the number of qualifications could be unlimited, we might not be able to express the situation at all.

We use a representation that makes event variables the central component for organizing knowledge about events. In particular, events are divided into types, and each type defines a set of role functions that specify the arguments to any particular event instance. The event of Jack lifting the ball onto the table at time $t1$ would be represented as

$$\exists e \,.\, \textsc{Lift}(e) \wedge agent(e) = jack34 \wedge dest(e) = table5 \wedge theme(e) = ball26 \wedge time(e) = t1.$$

Event predicates will always be written in SMALL CAPS to distinguish them from other functions and predicates.[1]

[1] The logic containing reified events also allows one to discuss events that do not occur. In particular, asserting that an event instance exists does not necessarily entail that it occurred. A new predicate *Occurs* can be introduced and the assertion that Jack lifted the ball (at time $t1$) would be represented as $\exists e \,.\, \textsc{Lift}(jack34, ball26, t1, e) \wedge Occurs(e)$. Using an explicit *Occurs* predicate allows events to exist even if they do not actually occur, or could never occur. We will not need this expressive power in this paper so will proceed under the interpretation that only events that occur exist. Not having to put the *Occurs* predicate in each formula will simplify the presentation.

This representation is somewhat verbose for presenting examples. When needed to make a point, the representation using only functions on events will be used. At other times, however, we will use the more standard predicate-argument notation as a convenient abbreviation. Thus, we will usually abbreviate the above formula as:

$$\exists e\,.\,\textsc{Lift}(jack34, ball26, table5, t1, e).$$

The arguments in this predicate-argument form will depend on the predicate, but the last two argument positions will always be the time of the event and the event instance, respectively. Finally note that event predicates are anti-homogeneous (that is, they hold over no sub-interval of the time over which they hold) as discussed in Section 2.1.

Because of the representation based on role functions, an event instance uniquely defines all its arguments. This is important to remember when the predicate-argument abbreviation is used. In particular, if we asserted that both $\textsc{Lift}(a_1, b_1, c_1, t_1, e)$ and $\textsc{Lift}(a_2, b_2, c_2, t_2, e)$ were true, then this would entail that $a_1 = a_2$, $b_1 = b_2$, $c_1 = c_2$, and $t_1 = t_2$, a fact not obvious when using the predicate-argument form but clear from the functional form.

We will represent knowledge about events in several ways. The first is by defining necessary conditions on the event occurring. For instance, consider the event of one block being stacked on another by a robot. This could be described by an event predicate of form $\textsc{Stack}(x, y, t, e)$. Axioms then define the consequences of this event occurring. For instance, one might assert that whenever a stack event occurs, the first block is on the second block at the end of the action. In the predicate-argument notation, this could be written as

$$\forall x, y, t, e\,.\,\textsc{Stack}(x, y, t, e) \supset \exists t'\,.\,t : t' \wedge On(x, y, t').$$

In the functional form, the same axiom would be

$$\forall e\,.\,\textsc{Stack}(e) \supset \exists t'\,.\,time(e) : t' \wedge On(block1(e), block2(e), t').$$

Of course, there are many other necessary conditions in order for a stacking event to occur. For instance, we might say that the block moved ($block1(e)$) must be clear when the event starts, that the agent must be holding that block some time during the event (actually up to the end of the event), and that the other block ($block2(e)$) is clear just before the end of the event, and has *block1* on it immediately after the event completes. The event terminates at the time when *block1* is on *block2*. This information can be expressed directly using the temporal logic by the following axiom:

$$\begin{aligned}
&\forall x, y, t, e\,.\,\textsc{Stack}(x, y, t, e) \supset \\
&\quad \exists j, k, l, m, n\,.\,Clear(x, j) \wedge Overlaps(j, t) \wedge \\
&\qquad Holding(x, k) \wedge Finishes(k, t) \wedge j : k \wedge \\
&\qquad Clear(x, l) \wedge t : l \wedge Clear(y, m) \wedge SameEnd(t, m) \wedge \\
&\qquad On(x, y, n) \wedge t : n.
\end{aligned}$$

A more useful form of this axiom for planning uses temporal functions on each event that define the structure of the temporal intervals needed for its definition. For example,

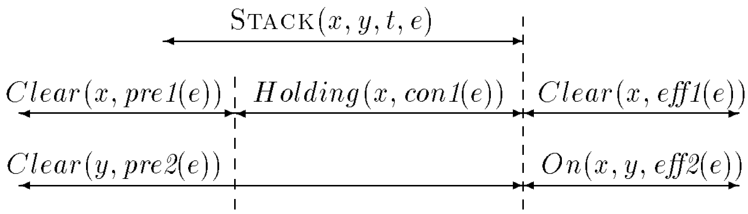

Figure 5: Necessary conditions for stacking x on y

for the class of stacking events, we need functions to produce times corresponding to the existential variables in the axiom given above. Using new function names, we might define the temporal structure of the stacking event as follows:

$$\begin{aligned}&\forall t, e \,.\, \textsc{Stack}(t, e) \supset \\ &\quad Overlaps(\mathit{pre1}(e), t) \wedge Finishes(\mathit{con1}(e), t) \wedge \mathit{pre1}(e) : \mathit{con1}(e) \wedge \\ &\quad t : \mathit{eff1}(e) \wedge SameEnd(t, \mathit{pre2}(e)) \wedge t : \mathit{eff2}(e).\end{aligned}$$

The temporal functions are named to informally suggest the three classes of conditions that arise in an event definition. The "preconditions"—conditions that must hold prior to the event's occurrence—have the prefix *pre*, the "effects"—conditions that must hold following the event—have the prefix *eff*, and the other conditions that must hold during the event have the prefix *con*. The temporal functions can be viewed as Skolem functions justified by the original axiom. The only aspect that will seem puzzling is that we will use the same Skolem functions in several axioms. This should be viewed as a notational convenience that allows a large conjunctive axiom (with a single universally quantified event variable) to be presented as several smaller axioms. With this temporal structure defined for every stacking event, the axiom defining the necessary conditions for the event's occurrence now can be expressed as follows:

$$\begin{aligned}&\forall x, y, t, e \,.\, \textsc{Stack}(x, y, t, e) \supset \\ &\quad Clear(x, \mathit{pre1}(e)) \wedge Holding(x, \mathit{con1}(e)) \wedge Clear(x, \mathit{eff1}(e)) \wedge \\ &\quad Clear(y, \mathit{pre2}(e)) \wedge On(x, y, \mathit{eff2}(e)).\end{aligned}$$

This axiom is shown graphically in Figure 5.

The above axiom asserts what is true whenever a stacking event occurs, independent of the situation. Other knowledge about events is relevant only in certain situations (*i.e.*, the event has conditional effects). For instance, if the block being moved in a stacking action was initially on another block, then this other block becomes clear. This is expressed in the logic by the following axiom, which states that if block x was initially on another block z, then z becomes clear when x is moved:

$$\begin{aligned}&\forall x, y, z, t, e, t' \,.\, \textsc{Stack}(x, y, t, e) \wedge On(x, z, t') \wedge Overlaps(t', t) \supset \\ &\qquad \exists T, t'' \,.\, t' \sqsubseteq T \wedge On(x, z, T) \wedge T : \mathit{con1}(e) \wedge Clear(z, t'') \wedge T : t''.\end{aligned}$$

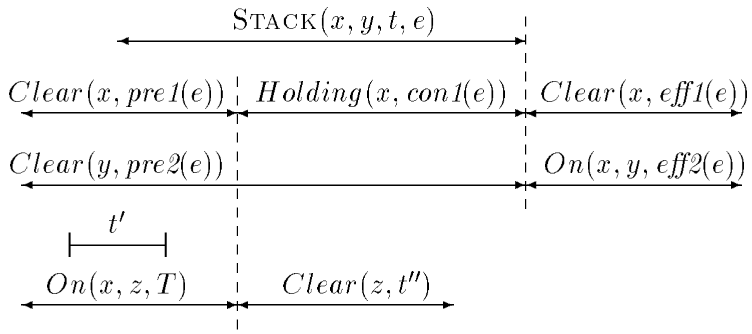

Figure 6: Conditional effects of stacking when x is initially on z

This situation is illustrated graphically in Figure 6. This axiom applies in a situation with three blocks, such as A, B, and C, where A is originally on block C. Note that this definition does not assert that block C will be clear at the end of the stacking event. In particular, if two stacking events overlap in time (say, $\textsc{Stack}(A, B, t1, e1)$ and $\textsc{Stack}(D, C, t2, e2)$), then this may not be the case, for D may be placed on C before A is placed on B. Most other representations cannot easily represent such subtleties.

It is worth noting that the conditional effect axiom above, although complicated by the interval-based ontology, does in fact demonstrate a typical pattern. Notice that the left-hand side of the implication contains the proposition $On(x, z, t')$, while the right-hand side contains $On(x, z, T)$ where $t' \sqsubseteq T$. The new interval T defines the extent of the property given that the event has occurred. While this formulation seems a bit cumbersome, it occurs with every conditional effect. Thus it would be easy to define an abbreviation convention to make specifying such axioms more convenient.

Of course, if the only events we needed to represent were simple events such as occur in the blocks world, then the temporal logic would be overkill. But we are interested in much more realistic events, where we may have significant knowledge about how the world changes as the event occurs. As a quick example, consider the event of a cup filling up with water. At the end the cup is full, but while the event is occurring, the level of the water is continually increasing. Commonsense knowledge about this event is easily captured in the interval logic. For instance, the fact that the level of the water continues to rise throughout the event is captured by the following axiom:

$$\begin{aligned}
&\forall c, e, t, t', t'' \,.\, \textsc{Fill}(c, t, e) \supset \\
&\quad \forall t, t'' \,.\, (t' \sqsubset t) \wedge (t'' \sqsubset t) \wedge (t' \prec\!: t'') \supset \\
&\qquad level(c, t') < level(c, t''),
\end{aligned}$$

where $level(c, t)$ is a function that gives the minimum level of the water in the cup over time t. This axiom captures the intuition that the water continually rises quite simply and directly without making a commitment to a continuous model of change. Dealing with events that involve such "continuous change" is not the focus of this paper, however, and most of our examples will involve the very simple actions that are common in the literature on planning and reasoning about action.

Finally, some might wonder why the logic has no separate mechanism for describing processes. While this may ultimately be necessary, the existing formalism already supports a wide range of "process-like" predicates. In particular, you can define a property that is true only when an event is occurring. In the above example about the cup, we could define a predicate $CupFilling$ as

$$\forall c, t \,.\, CupFilling(c, t) \equiv \exists t', e \,.\, t \sqsubseteq t' \wedge \textsc{Fill}(c, t', e)$$

Note that by this definition, $CupFilling$ is a homogeneous predicate, as expected for properties. The place where problems arise with this simple approach is in dealing with the imperfective paradox. For instance, you might want to say that a person was crossing the street at some time t, even if they changed their mind at the last minute and went back to the original side. This problem has been studied extensively in the philosophical literature, but has not been a focus for our work as it does not seem to arise naturally in the planning domains we have studied.

3.4 The Logic of Actions

The representation of events described in the previous section cannot adequately capture knowledge of causality. In particular, the formulas above do not state what properties are caused by the stacking action or what properties simply must be true whenever the action succeeds. This is the distinction that STRIPS makes between preconditions and effects. Intuitively, it is evident that the stacking action causes block A to be on block B in situations where both blocks are clear at the start of the action. Furthermore, the stacking action causes block B to become not clear while it does not affect the condition that block A is clear.

To encode such knowledge, we need to be able to reason about attempting to perform an action. The logic developed so far can express the fact that a certain event occurred, but not that an agent attempted to perform some action. Until we make such a distinction, we will not be able to explicitly describe the conditions under which an action can be successfully executed, or describe what happens when an action is tried in inappropriate circumstances. To do this, we need to better define the distinction between events and actions. So far, we have only talked about events occurring. The assertion that Jack lifted the ball onto the table describes an event in which Jack performed some action that resulted in the ball being lifted onto the table. The action Jack performed, namely, the lifting, corresponds to some set of motions that Jack performed in order to lift the ball. If Jack were a robot, the action would be the execution of a program that involved the correct control sequences given perceptual input. Thus, in a robot world the action corresponds to the program, whereas the event corresponds to a situation in which the program was executed successfully.

As noted in Section 2.1, for every action there is a corresponding event consisting of an agent performing that action. We will often exploit this by using the same names for events and actions. Thus the $\textsc{Stack}(x, y, t, e)$ predicate presented in the previous section might correspond to the action term $stack(x, y)$ that denotes a program where x and y correspond to the blocks being stacked. Of course, there are other events that do not involve actions. For example, natural forces (such as the wind blowing) result in events but do not involve action in the sense we are using it. Actions may be arbitrarily complex activities, and can be decomposable into other less complex actions, which themselves may be decomposable, until a certain basic level of action is attained. The primitive actions are called the *basic* actions following [Goldman, 1970].

The predicate Try is defined on programs, such that $Try(\pi, t)$ is true only if the program π is executed over time t.[2] As with event predicates, Try is anti-homogeneous. We can now assert an axiom defining the conditions sufficient to guarantee successful action attempts. In the case of stacking, whenever the agent tries to stack x on y starting in a situation where x and y are clear, then a stacking event occurs that is temporally constrained by the initial conditions:

$$\forall x, y, t, j, k \,.\, Try(stack(x, y), t) \wedge Clear(x, j) \wedge Overlaps(j, t) \wedge \\ Clear(y, k) \wedge SameEnd(k, t) \supset \exists e \,.\, \textsc{Stack}(x, y, t, e)$$

Of course, a realistic axiom would also include duration constraints on t so that the action is attempted sufficiently long enough to allow it to succeed, *etc.*.

Note that the logic does not presuppose any particular relationship between when the action is performed and when the event caused occurs. The only constraint we need, based on commonsense notions of causality, is that the caused event cannot precede the action that caused it. But otherwise, they can be simultaneous, or the event might be included within the time of the action performance, or it might immediately follow the action. In practice, there are two forms of axioms that are most common and useful. In *simultaneous* causality, the action and the event caused are simultaneous, as in the above example in which the stacking event is co-temporal with the action execution (recall that the abbreviation $\textsc{Stack}(a, b, t, e)$ implies $t = time(e)$). With *pushbutton* causality, the event immediately follows the action, say as the action of pushing a ball causes it to roll down a hill. Note that the issue of how the action relates to the event is independent of how an event relates to its effects, although the same distinctions can be made. For example, the effect of stacking two blocks starts holding at the end of the event. The effect of holding a door shut, on the other hand, might only hold while the event is occuring. Which relationships between actions, events and effects, is best to use for any particular problem will depend on one's intuitions about the domain, and the level of detail at which one needs to reason about the domain.

It is important to consider why the information about stacking is captured by two different sets of related axioms: one capturing the necessary conditions whenever a stacking event occurs (Section 3.3), and the other relating action attempts to event occurrences (above). This is because the two sets of axioms represent two very different sources of knowledge. The first defines knowledge about what the world is necessarily like whenever the event occurs successfully, while the second defines the abilities of the agent in causing

[2] In previous work (*e.g.*, [Allen, 1991]), we included the event "caused" by attempting the action as an argument to the Try predicate. The problem with this is that we might want to categorize the "caused" event in several ways (*i.e.*, using several event predicates).

events. In many situations, an agent may know the former but not the latter. For example, we all can recognize that the mechanic fixed our car, even if we have no idea what enabled the mechanic to do the job. In this case, we have knowledge of what it means to fix a car as expressed by the following axiom, which states that fixing something means that the thing was once broken and is now working:

$$\begin{aligned}&\forall c, t, e \,.\, \textsc{Fix}(c, t, e) \supset \\ &\quad Broken(c, \mathit{pre1}(e)) \wedge Finishes(t, \mathit{pre1}(e)) \wedge \\ &\quad Working(c, \mathit{eff1}(e)) \wedge Meets(t, \mathit{eff1}(e)).\end{aligned}$$

With this information, the agent knows what it means for the car to be fixed (although the agent does not know how it was done). Furthermore, such axioms could be used by a system to recognize situations in which the car has been fixed. Knowledge of this sort is also essential for much of natural language semantics, where many verbs are defined and used without the agent's knowing the necessary causal knowledge. Allen [1984] discusses this at length.

To make this distinction concrete, we can partition our axioms describing events and actions into three broad categories:

EDEF Event definitions – These are axioms of the form

$$\forall e \,.\, E(e) \wedge \phi \supset \psi,$$

where E is an event-type predicate and ϕ and ψ contain no event predicates. The necessary conditions for stacking given in the previous section fall into this category.

ETRY Action definitions – These are axioms of the form

$$\forall t, \ldots \,.\, Try(\pi, t) \wedge \phi \supset \exists e \,.\, E(e) \wedge t \circ time(e) \wedge \psi,$$

where again E is an event predicate. In this case, ψ represents constraints on e, and can involve other quantified variables. The symbol "$\circ$" stands for the temporal relation between the action and the event, typically "*Equal*" or "*Meets*", as discussed above.

EGEN Event generation axioms – These are of the form

$$\forall e \,.\, E(e) \wedge \phi \supset \exists e' \,.\, E'(e') \wedge \psi.$$

Again, E and E' are event-type predicates and ϕ and ψ represent constraints on the events. The classic example of event generation is represented in English using the "by" locution, for example, signaling a turn by waving one's arm, under appropriate conditions. More concrete examples will be presented in later sections.

Examples of all three classes of axioms will be presented in the next section when we describe application of the formalism to a set of problems from the literature on reasoning about action.

3.5 Discussion

Having introduced our representation of time, actions and events, we can now compare it to the other major formalisms. Note that this comparison is based on the expressiveness and simplicity of the logic for capturing commonsense knowledge. Dealing with prediction is a separate issue for most of these formalisms, and will be discussed at the end of the next section.

As discussed previously, most formalisms have not included an explicit model of time, but base their model on states or situations. To handle explicit temporal relations in the situation calculus, a function can be defined that maps a state or situation to a timepoint. In this way, the situation calculus defines a point-based branching time model. A duration function on actions can be defined that allows one to compute the time of the resulting situation given the initial situation [Gelfond *et al.*, 1991]. Within the constructive models, however, the range of temporal reasoning that can be performed is severely limited. First, note that there are no situations defined during the time an action is executing. This means that you cannot assert anything about the world during an action execution. Rather, all effects of an action must be packaged into the resulting situation. As a consequence, all effects start simultaneously. Since situations correspond to time points, the formalism also has no mechanism for representing knowledge about how long an effect might hold, as time only moves forward as a result of an action. The only way around this seems to be to allow null actions that move time forward but have no effect, which is neither convenient or intuitive. In the temporal logic with events, you have the full power of the formalism for representing information about the temporal properties of events and their effects. In addition, effects P and Q of some event are each defined to hold over separate intervals. These intervals are independent of each other, and so we can reason about how long P remains true without worrying about how long Q holds.

As a consequence, the only worlds easily represented by the constructive situation calculus are those that remain static except when the agent acts, and where nothing important happens while actions are being executed. Of course, if one abandons the requirement of a constructive theory, then more complicated scenarios can be represented. In fact, in some approaches (*e.g.*, [Shanahan, 1990]) even continuous change has been represented. The primary mechanism that enables this, however, is the exploitation of the mapping from situations to times, and the introduction of situations that are not the result of any actions. Given these radical extensions, it is not clear to us what the advantage is in retaining the situation calculus as the underlying theory, rather than moving to the simpler, more direct, explicit temporal logic. Temporal logic gives properties, actions and events equal temporal status, allowing complex situations to be captured quite directly in a manner similar to how they are intuitively described.

We have also made a distinction between actions and events that is not made in most other formalisms, which use only one construct. So an interesting question is whether these other representations involve actions or events? This question is hard to answer for STRIPS-based systems, as they say nothing about unsuccessful attempts of actions. A STRIPS-style system, for instance, only gives information about the effect of an action when the preconditions holds (*i.e.*, when the action is guaranteed to succeed). There need be no distinction made between actions and events in these circumstances, as there is a one-to-one mapping between actions and the events that they cause.

The situation calculus, on the other hand, does allow multiple axioms defining the consequences of an action, so the same action does not always have the same effect. This seems close to our intuitions about actions. For instance, the situation calculus allows the following axioms, which state how the stacking action might result in different situations when performed in different circumstances:

> If a and b are clear in situation s, then $On(a, b)$ is true in situation $Result(stack(a, b), s)$.
> If a is clear, and $On(c, b)$ is true in situation s, then $On(a, c)$ is true in situation $Result(stack(a, b), s)$.

There is no explicit representation of events in the situation calculus, which makes it difficult to represent knowledge of the actual world. For instance, it is not possible to directly assert that an event E will occur tomorrow, or even that the agent will perform an action A tomorrow. In the constructive situation calculus, one would have to quantify over all action sequences and ensure that the event or action was present. This does not seem workable, and recent attempts to represent the actual events generally abandon the constructive approach, and have to introduce an explicit new construct for events (*e.g.*, [Miller and Shanahan, 1994]). The resulting formalism is again closer to what we propose, and we question the advantage of retaining the original situation calculus framework.

The event calculus [Kowalski and Sergot, 1986] has an explicit notion of event that corresponds more closely to our notion of events. This is not surprising as both this formalism and our own development of the temporal logic are based on intuitions about the close relationship between time and events. The event calculus does not seem to have a separate notion of actions, although formulas of the form $act(E, give(a, x, y))$ are used that suggest they might. As far as we can tell, however, this is only a syntactic mechanism for naming events. Reasoning in the event calculus is driven by the events that occur, as it is in our approach, but the relationship between the behaviors that an agent might perform and the events that the agent causes is not explored. Their representation of events seems weaker than what we are proposing. Specifically, the formalism only allows the simplest temporal relationships between an event and its effects (in our terms, all effects are "$MetBy$" the event), and it is not clear how events could be extended in time. The most significant difference, however, is that the event calculus formalism is organized around a specific strategy for dealing with the frame problem. Specifically, it uses a negation as failure to implement a persistence strategy where a proposition remains true until an event explicitly changes (or clips) it. For this strategy to be valid, assumptions must be made that are similar to the STRIPS assumptions.

Closest to our representation is that described by McDermott [1985]. He does distinguish between actions (which he calls tasks) and events, and uses an explicit model of time. As a result, he can define notions such as a successful action attempt, and explicitly reason about an agent's actions and what events they will cause. His representation of time has many similarities to the work that extends the situation calculus with a time line [Pinto, 1994], and he explores the consequences of adding explicit time to such a model in depth.

A significant difference between our approaches is that we use an interval-based logic, while McDermott uses a continuous point-based model. These differences have been discussed in detail elsewhere (*e.g.*, [van Benthem, 1983; Allen, 1984; Shoham, 1988]).

4 Reasoning about Action in Simple Domains

This section discusses prediction in theories of action, in particular the frame problem, and describes our approach based on explanation closure. The approach is illustrated by formalizing and solving a set of simple problems from the literature on reasoning about action. Extensions of the theory to more complicated and realistic domains are presented in subsequent sections.

4.1 Prediction

As mentioned in the introduction, there are three major reasoning tasks we require the logic of events and actions to support: prediction, planning, and explanation. Of these, prediction is the most basic capability—given a description of the world and a set of actions and events that will occur, what can we conclude about the past, present, and future? With a temporal logic, the initial world description might already contain some information about the past or the future, say some external events that will occur, or future actions that the agent knows it will perform. The prediction task reduces to predicting the effects of new actions and events that are posited to occur, and updating the world model accordingly. Neither planning nor explanation can be accomplished without a model of prediction. In planning, for instance, the task is to find a set of actions that will accomplish a given set of goals. This can be divided into two abstract tasks: generating a candidate set of actions, and evaluating whether the plan will succeed. This latter task is exactly the prediction task. Explanation can be similarly decomposed into generating a possible set of events that might explain the observations, and then verifying whether the events would actually cause the observed effects. Of course, any particular algorithm might not divide the reasoning into two explicit steps. In fact, most planning algorithms exploit a specific prediction model in order to suggest actions likely to produce a good plan, but all systems are based on some prediction model.

As a concrete example, consider the standard backward chaining planning algorithm using the STRIPS representation (*e.g.*, [Nilsson, 1980]). The algorithm chains backwards from the goal state. First, the goal state is compared to the initial state and a set of propositions that differ in truth value between the two states are found. Then, one of these propositions is chosen and an action is introduced that has the proposition as one of its effects. Then the state of the world before this action is performed is computed using a technique called regression, which essentially inverts the add and delete lists in the action's definition. This new state now becomes the goal state and the process continues until the initial state is derived. Once that is done, the algorithm has found a sequence of actions leading from the initial state to the goal state as desired. The prediction model that makes this approach valid includes the following assumptions:

- No other events or changes occur in the world except for the planned actions;
- The action definitions completely describe all changes that occur as the result of the action.

With these two assumptions, prediction is accomplished simply by applying the transformations defined by each of the actions. The regression technique was designed so that an explicit prediction step was not necessary, given the assumptions. Specifically, in regressing an operator back from a state s to a preceding state s', one is guaranteed that predicting from s' with the same action would yield back s. By exploiting this property, the plan can be constructed in this backward fashion, and once a plan is found, it is guaranteed to achieve the goal given the two assumptions above.

The same prediction model underlies the formalisms based on non-linear planning as in TWEAK [Chapman, 1987] and SNLP [McAllester and Rosenblitt, 1991] and systems based on these techniques. The additional complication is that actions are partially ordered, and so systems must distinguish between predictions that are necessarily true (in any allowable action ordering) or only possibly true (in some allowable action orderings). With this addition, the new prediction model validates the proposed search strategies. Similar methods using the event calculus are proposed in [Eshghi, 1988; Shanahan, 1989; Kowalski, 1992].

4.2 Frame Axioms and Explanation Closure

The problem with the above approach is that the STRIPS assumptions do not hold in most realistic situations that one needs to reason about. The situation calculus and temporal logic-based approaches do not have to operate with such assumptions. As a result, these theories themselves make little commitment to how the state resulting from an action relates to the state before the action. Rather the properties of the resulting state must be specified axiomatically, and the frame problem involves how best to specify these properties. The original proposal for the situation calculus [McCarthy and Hayes, 1969] was to use *frame axioms*, which explicitly stated which properties are not changed by the actions. Explicit frame axioms have come under criticism, primarily because there are too many of them, both to write down explicitly and to reason with efficiently.

There are several ways to try to overcome this problem. Many researchers abandon frame axioms altogether, and have built models that use persistence or inertia assumptions (*e.g.*, [Lifschitz, 1987; Shoham, 1988]). These approaches assume that all changes caused by an action are specified, and every property not asserted to change does not change. This technique has much to recommend it, as it eliminates the need to enumerate frame axioms, but in its simple form it tends to be too strong. In particular, if there is uncertainty as to whether a property might change or not, techniques based on this approach will often incorrectly assume that the change does not occur. Other approaches work instead by minimizing event or action occurrences. Properties are assumed to change only as a result of events defined in the representation, and logically unnecessary events do not occur (*e.g.*, [Georgeff, 1986b; Georgeff, 1986a; Morgenstern and Stein, 1988]). These approaches show more promise at handling more complex situations and have many similarities to our work.

The approach we take, however, retains the flavor of explicit frame axioms. Rather than specifying for each action whether each property changes or not, however, one specifies for each property what events can change it. The problem of reasoning about changes then reduces to the problem of reasoning about what events may or may not have occurred. This technique is called the *explanation closure* approach and was proposed by Haas [1987] and Schubert [1990]. Schubert has shown that such a technique can dramatically reduce the number of frame axioms required to produce a workable set of axioms for a problem. Of course, assumptions must still be made. As in other approaches, we assume that unnecessary events do not occur, but this is specified axiomatically rather than being built into the semantic model. This has several advantages. Most importantly, the resulting axioms can be interpreted with the standard semantics of the first-order predicate calculus, meaning that there are well-defined notions of entailment and proof. We can show that our representation handles a particular class of examples by showing a proof of the desired consequences, without needing to appeal to model-theoretic arguments in a non-standard semantics. In addition, various forms of uncertainty are handled using the standard methods in logic with disjunction and existential quantification. Finally, the assumptions that are made appear explicitly as axioms in the system. While not exploited in this paper, this allows for explicit reasoning about the assumptions underlying a line of reasoning (*cf.* [Ferguson and Allen, 1994; Ferguson and Allen, 1993; Allen, 1991; Ferguson, 1992]).

If you are willing to reinstate strong assumptions about the domain, roughly equivalent to the STRIPS assumptions, then Reiter [1992] has shown that the explanation closure axioms can be computed automatically using predicate completion techniques. But this close correspondence breaks down in more complex domains. Schubert [1990] argues that explanation closure is distinct from predicate completion or biconditionalization, and thus that the axioms cannot be generated automatically. Specifically, he argues that any automatic procedure will produce axioms that are too strong in cases where knowledge of the world is incomplete and actions may have conditional effects. In addition, he argues that explanation closure axioms have "epistemic content" and are independently motivated by other problems such as language understanding. As such, they form a crucial body of knowledge necessary for many commonsense reasoning tasks. Notwithstanding this issue, it seems likely that explanation closure axioms can be generated automatically in many cases, for instance for actions that never have conditional effects. But this is a programming issue rather than a logical issue. As far as the formalism is concerned, the closure axioms are an essential part of the reasoner's knowledge of the domain are are not automatically derivable. As part of a practical knowledge representation, all sorts of techniques could be developed to make the specification of this knowledge easier, but these are not the subject of this paper.

The examples to follow in the rest of this section do not demonstrate the full power of our approach, as they are formulated not to involve simultaneous actions and external events. But they will facilitate the comparison of our work to other approaches, and the more complex issues will be considered in later sections. Because of the simple nature of the problems, the closure axioms can be classified into two classes, corresponding to two assumptions: no properties change unless explicitly changed by an event occurring, and no events occur except as the result of the actions. Although these are difficult to precisely describe schematically, they look something like the following following:

EXCP (Strong Closure on Properties) Every property change results from the occurrence of an instance of one of the event types defined in the axioms. These axioms are of the form:

$$\forall t, t' \,.\, P(t) \wedge \neg P(t') \wedge t : t' \supset \exists e \,.\, (E_1(e) \vee E_2(e) \vee \ldots) \wedge time(e) : t',$$

where the E_i are the event-type predicates that (possibly) affect the truth value of P. These axioms are derived from the event definition axioms (EDEF).

EXCE (Strong Closure on Events) Every event that occurs does so as a result of some action being attempted, possibly indirectly via event generation. These axioms are of the form: "$\forall e \,.\, E(e) \supset \phi_1 \vee \phi_2 \vee \ldots$," where ϕ_i is either of the form

$$\exists t \,.\, Try(\pi, t) \wedge t \circ time(e) \wedge \psi,$$

for π an action term and "$\circ$" a temporal relation (typically "$=$"), or of the form

$$\exists e' \,.\, E'(e') \wedge time(e) \circ time(e') \wedge \psi,$$

for E' an event-type predicate. These axioms can often be derived from the action definition (ETRY) and event generation (EGEN) axioms.

4.3 Examples

By way of illustrating the application of the logic and the explanation closure technique, we now present formalizations of some of the standard problems from the literature on reasoning about action gathered in [Sandewall, 1994]. These are mostly variants of the Yale Shooting Problem [Hanks and McDermott, 1986], a basic AI test case for reasoning about action. Each scenario involves a very simple domain where a single problem is presented and thus it is not the best vehicle for demonstrating the generality of our approach. However, it does establish a certain baseline and allows us to support our claims as to the expressiveness and naturalness of the event-based temporal logic.

To begin, consider our formulation of the basic Yale Shooting Problem (YSP). The scenario involves a hapless victim, Fred, who apparently sits idly by while an adversary loads a gun, waits some period of time, and shoots (at Fred). The question is whether Fred is dead or not, or rather whether an axiomatization of the problem in some logical system allows the conclusion that Fred is dead to be derived. This simple problem has generated an extensive body of literature (*cf.* [Brown, 1987]). In most cases, the issue is that while the loadedness of the gun ought to persist through the waiting (and then the shooting succeeds in killing Fred), Fred's "aliveness" ought not to persist through the shooting (indeed, cannot, without risk of inconsistency) although it should persist through the loading and waiting. On the other hand, there might be "non-standard" models where the gun somehow becomes unloaded, in which case the shooting would fail to kill Fred.

Figure 7 shows the axioms for the YSP in our logic; they are depicted graphically in Figure 8. Recall that the EDEF axioms describe the necessary conditions for event occurrence

EDEF1 $\forall e\,.\,\textsc{Load}(e) \supset \mathit{Loaded}(\mathit{eff1}(e)) \wedge \mathit{time}(e) : \mathit{eff1}(e)$

EDEF2 $\forall e\,.\,\textsc{Shoot}(e) \supset \mathit{Loaded}(\mathit{pre1}(e)) \wedge \mathit{SameEnd}(\mathit{pre1}(e), \mathit{time}(e)) \wedge \neg \mathit{Loaded}(\mathit{eff1}(e)) \wedge \mathit{time}(e) : \mathit{eff1}(e) \wedge \neg \mathit{Alive}(\mathit{eff2}(e)) \wedge \mathit{time}(e) : \mathit{eff2}(e)$

ETRY1 $\forall t\,.\,\mathit{Try}(\mathit{load}, t) \supset \exists e\,.\,\textsc{Load}(t, e)$

ETRY2 $\forall t\,.\,\mathit{Try}(\mathit{shoot}, t) \wedge \mathit{Loaded}(t) \supset \exists e\,.\,\textsc{Shoot}(t, e)$

Figure 7: Basic axioms for the Yale Shooting Problem

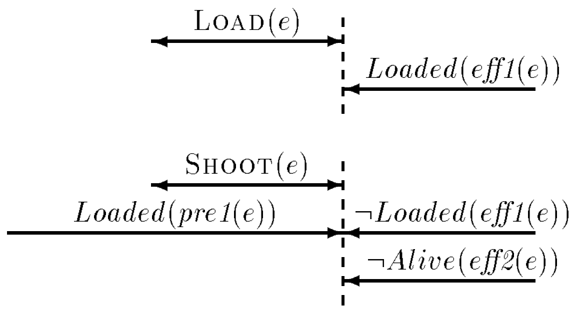

Figure 8: The LOAD and SHOOT event definitions

EXCP1 $\forall t, t' . \neg Loaded(t) \wedge Loaded(t') \wedge t : t' \quad \supset \quad \exists e . \textsc{Load}(e) \wedge time(e) : t'$

EXCP2a $\forall t, t' . Loaded(t) \wedge \neg Loaded(t') \wedge t : t' \quad \supset \quad \exists e . \textsc{Shoot}(e) \wedge time(e) : t'$

EXCP2b $\forall t, t' . Alive(t) \wedge \neg Alive(t') \wedge t : t' \quad \supset \quad \exists e . \textsc{Shoot}(e) \wedge time(e) : t'$

EXCE1 $\forall e . \textsc{Load}(e) \quad \supset \quad \exists t . Try(load, t) \wedge t = time(e)$

EXCE2 $\forall e . \textsc{Shoot}(e) \quad \supset \quad \exists t . Try(shoot, t) \wedge t = time(e)$

Figure 9: Explanation closure axioms for the YTS

and the ETRY axioms describe the sufficient conditions relating program executions (actions) and events. Of course, the example axioms are very simple—there are only temporal roles and there are no generation axioms (yet).

Figure 9 gives the explanation closure axioms corresponding to the event and action axioms in Figure 7. Given the simple actions and strong assumptions made in the problem formulation, an automatic procedure could probably devised to generate them. But notice that the closure axioms are not simply a strengthening of the implication to a biconditional. In particular, the EXCP2b axiom does not say that every SHOOT event causes a transition from alive to dead, just that every such transition is the result of a SHOOT. This would allow one to shoot things that are already dead, for example.

In the remainder of this section, we use these axioms for several of the core problems of the Sandewall test suite [Sandewall, 1994]. Each of these problems deals with prediction from a given situation, sometimes predicting previous facts (retrodicting). For each problem, we provide an axiomatization of the problem description and a proof (or proof sketch) of the conclusion. The role of Fred is taken by a turkey in the Sandewall collection, hence the change in terminology.

Yale Turkey Shoot (YTS)

This is the original YSP scenario described above. Initially the turkey is alive and the gun is not loaded. The agent loads the gun, waits, and shoots. We are to show that the turkey is dead sometime after the shooting. At issue is the persistence of loadedness through the waiting and the non-persistence of aliveness through the shooting. The following axioms describe the scenario:

AX0 $t0 \prec t1 \prec t2 \prec t3$

AX1 $Alive(t0)$

AX2 $\neg Loaded(t0)$

AX3 $Try(load, t1)$

AX4 $Try(shoot, t3)$

AX5 $\forall a, t . Try(a, t) \equiv (a = load \wedge t = t1) \vee (a = shoot \wedge t = t3)$

The final axiom is the important assumption that we attempt only the given actions. With the EXCE axioms, this ensures that "nothing else happens," which drives the explanation closure method. Note that the formalism does not require such an assumption. Rather, we are making explicit an aspect of the scenario that is usually implicit and accomplished via some form of semantic minimization.

To Prove: $\exists t \,.\, \neg Alive(t) \wedge t3 \prec: t$

Proof: Since there are no preconditions for loading, AX3 and ETRY1 give

$$\exists e_1 \,.\, \textsc{Load}(t1, e_1),$$

then EDEF1 gives

$$Loaded(\mathit{eff1}(e_1)) \wedge t1 : \mathit{eff1}(e_1).$$

That is, the loading succeeds, and the gun is loaded, at least immediately afterwards. Now, suppose that

$$Loaded(t3) \qquad\qquad (*)$$

that is, the gun remains loaded until the shooting is attempted. In that case, AX4 and ETRY2 give $\exists e_3 \,.\, \textsc{Shoot}(t3, e_3)$, and then EDEF2 gives

$$\neg Alive(\mathit{eff2}(e_3)) \wedge t3 : \mathit{eff2}(e_3).$$

That is, the shooting also succeeds and the turkey is dead afterwards, as required.

To show the persistence $(*)$, suppose otherwise that $\neg Loaded(t3)$. The interval temporal logic theorem DISJ (Section 3.2) gives $\mathit{eff1}(e_1) \bowtie t3$, then AX0 and the fact that $t1 : \mathit{eff1}(e_1)$ give $\mathit{eff1}(e_1) \prec: t3$. We apply interval temporal logic theorem TRPT to get

$$\exists T, T' \,.\, Loaded(T) \wedge \neg Loaded(T') \wedge T : T' \wedge t3 \sqsubseteq T'.$$

Then, EXCP2a gives

$$\exists e \,.\, \textsc{Shoot}(e) \wedge time(e) : T'.$$

That is, if the gun became unloaded it must have been because of a shooting. Note that $time(e) \prec: t3$ since $t3 \sqsubseteq T'$ and $time(e) : T'$. Then from EXCE2 we get

$$\exists t \,.\, Try(shoot, t) \wedge t = time(e),$$

i.e., if there was a shooting then someone must have tried to shoot. Since $t = time(e) \prec: t3$, we have $t \neq t3$, contradicting AX5. $\square$

Note that this proof will go through in exactly the same way even if you add other actions, such as waiting or eating lunch, at time $t2$, as long as the definition of such actions and their closure axioms indicate that they do not effect the gun being loaded. For example, the action of waiting for time to pass would be simply axiomatized as follows, assuming there were no preconditions for waiting:

ETRY3 $\forall t \,.\, Try(wait, t) \supset \exists e \,.\, \textsc{Wait}(t, e).$

Since the event has no effects, there is no need for an EDEF axiom and no need to change the EXCP axioms. There would, of course, be a corresponding EXCE axiom stating that waiting events only occur if the agent tries to wait. It should then be clear that adding $Try(wait, t2)$ to the axiomatization of the YTS scenario and adjusting AX5 accordingly will not affect the proof that the turkey is dead. Adding a more complicated action, such as eating lunch, with preconditions and effects would not be any different, provided it didn't affect the validity of the closure axioms (as it shouldn't if it is independent of loading and shooting).

As Schubert [1992] has remarked, the formulation based on explanation closure makes strong assumptions about the situation, as reflected in the closure axioms. In particular, the axiom AX5 that "nothing else happens" rules out many of the "problems" that arise in nonmonotonic approaches to this problem, such as ghosts appearing and unloading the gun during the interval between loading and shooting. Is this reasonable? It certainly seems more reasonable to make such an assumption explicit (as with AX5) and use a standard, well-understood logic than to build the assumptions into the semantics of the logic to make the intuitive solutions fall out as theorems. Such an assumption might be justified by the agent's knowledge of its plans, or by conventions about how stories are generally told, for example.

Stanford Murder Mystery (SMM)

In this variant, the turkey is initially alive but nothing is known about the state of the gun. The agent shoots and the turkey is observed to be dead after the shooting. We are to conclude that the gun was initially loaded. That is, we are required to "predict" aspects of the situation backwards in time (sometimes called retrodiction). This can be difficult for reasoners that "move" forward in time.

We have the following axioms:

AX0 $t0 \prec t1 \prec t2$

AX1 $Alive(t0)$

AX2 $\neg Alive(t2)$

AX3 $Try(shoot, t1)$

AX4 $\forall a, t \,.\, Try(a, t) \equiv (a = shoot \wedge t = t1)$

To Prove: $Loaded(t0)$

Proof: To start, AX0, AX1, AX2, and TRPT give

$$\exists T, T' \,.\, Alive(T) \wedge \neg Alive(T') \wedge T : T' \wedge t2 \sqsubseteq T',$$

then EXCP2b gives

$$\exists e \,.\, \textsc{Shoot}(e) \wedge time(e) : T'.$$

That is, a shooting event must have occurred between $t0$ and $t2$ to account for the death of the turkey. Then EDEF2 gives

$$Loaded(pre1(e)) \wedge SameEnd(pre1(e), time(e)),$$

i.e., since the shooting was successful, the gun must have been loaded.

Now, suppose $\neg Loaded(t0)$. Then DISJ, AX0, and the temporal constraints on $pre1(e)$ give $t0 \prec: pre1(e)$, then TRPT and EXCP1 give

$$\exists e' . \textsc{Load}(e') \wedge time(e') : T'',$$

where $pre1(e) \sqsubseteq T''$. That is, there must have been a loading event if the gun was initially unloaded and subsequently loaded. Applying EXCE1 gives

$$\exists t' . Try(load, t') \wedge t' = time(e'),$$

which contradicts AX4 as we know there was no loading attempted. Thus the gun must have initially been loaded, *i.e.*, $Loaded(t0)$. □

We see that the temporal logic is immune to the "direction" of the proof with respect to the direction of time, as expected. In contrast, most other models allow temporal prediction only in a forward direction (*e.g.*, [Dean and McDermott, 1987]) and must use a completely different approach for "persisting" backwards in time.

Interestingly, the proof for the SMM does not use the fact that we tried to shoot (AX3). Rather the facts that (a) the turkey died (AX1, AX2), (b) the only way it could have died was if we shot it (EXCP2b), and (c) we didn't try to do anything but shoot (AX4, in particular, we didn't load), were sufficient to prove the goal. The fact that we tried to shoot turns out to be a consequence of these.

Russian Turkey Shoot (RTS)

This variant introduces a new action, that of spinning the barrel of the gun, that randomizes its loadedness. At issue is the ability of the representation to deal with uncertainty in the effects of actions. The following axiom describes spinning the barrel:

ETRY3 $\forall t . Try(spin, t) \supset \exists e . \textsc{Spin}(t, e)$

There is no EDEF axiom for *Spin*, since the effect of spinning is the tautology $Loaded \vee \neg Loaded$. However, the explanation closure axioms (Figure 9) need to be modified to accommodate spinning:

EXCP1 $\forall t, t' . \neg Loaded(t) \wedge Loaded(t') \wedge t : t' \supset$
$\exists e . (\textsc{Load}(e) \vee \textsc{Spin}(e)) \wedge time(e) : t'$

EXCP2a $\forall t, t' . Loaded(t) \wedge \neg Loaded(t') \wedge t : t' \supset$
$\exists e . (\textsc{Shoot}(e) \vee \textsc{Spin}(e)) \wedge time(e) : t'$

EXCE3 $\forall e . \textsc{Spin}(e) \supset \exists t . Try(spin, t) \wedge t = time(e)$

In this case the explanation closure axioms include information not present in the original axioms, namely that spinning is a possible cause of the gun becoming unloaded, despite the lack of an EDEF axiom stating that it necessarily causes the gun to become unloaded.

The problem setup is the same as for YTS, but the agent loads, spins, and then shoots. We should now no longer conclude that the turkey dies. The axioms are the same as for the YTS with the addition of AXs that says the agent spins, and the modification of AX5 to allow spinning.

AX0 $t0 \prec t1 \prec t2 \prec t3$

AX1 $Alive(t0)$

AX2 $\neg Loaded(t0)$

AX3 $Try(load, t1)$

AXs $Try(spin, t2)$

AX4 $Try(shoot, t3)$

AX5 $\forall a, t \,.\, Try(a, t) \equiv$
$$(a = load \wedge t = t1) \vee (a = spin \wedge t = t2) \vee (a = shoot \wedge t = t3)$$

Proof: With these changes, the proof starts as for YTS, but in attempting to refute the hypothesis that the gun remains loaded $(*)$, we now get

$$\exists e \,.\, (\textsc{Shoot}(e) \vee \textsc{Spin}(e)) \wedge time(e) : T'$$

for $t3 \sqsubseteq T'$ from the modified EXCP2a. Then EXCE2 and EXCE3 give

$$\exists t \,.\, (Try(shoot, t) \vee Try(spin, t)) \wedge t = time(e),$$

which no longer contradicts the modified AX5. □

We see that the temporal logic with explanation closure can accommodate uncertainty in the effects of actions, so long as their potential occurrence is taken account of. Of course, the conclusions are weaker, but that is a consequence of the uncertainty inherent in the problem, not a shortcoming of the logic.

Dead Xor Alive (DXA)

For this variant, we replace $\neg Alive$ by $Dead$ as the effect of successful shooting, and add an axiom

$$\forall t \,.\, Dead(t) \equiv \neg Alive(t).$$

The logical equivalence leads to what Sandewall terms "autoramifications," but poses no problem for the deductive approach presented here.

Walking Turkey Problem (WTP)

Similarly to DXA, we are given that the turkey is known to be walking initially, but not explicitly known to be alive. We have the axiom

$$\forall t \,.\, Walking(t) \supset Alive(t),$$

however, and we are to conclude that the turkey is both dead and not walking after shooting. The proof of death is exactly like YTS after one application of the new axiom, and then the contrapositive of that axiom is used to show that the turkey is not walking once dead.

4.4 Discussion

In all the examples presented above, the central issue is how to reason formally about what changes and what doesn't change as the result of some action, *i.e.*, the frame problem. Work on the frame problem investigates methods of making assumptions about the world to predict the likely consequences of actions. There are therefore two separable issues to consider, which have been called the "epistemological" and "computational" aspects of the problem (*cf.* [Kowalski, 1992], but also [McCarthy and Hayes, 1969] regarding epistemological and heuristic adequacy). The epistemological aspect concerns what assumptions one makes about the world, while the computational aspect concerns how to compute and use these assumptions in the formalism. For example, it is an epistemological issue whether to make assumptions about property changes as in the persistence-based approaches, or to make assumptions about event occurrences. On the other hand, it is a computational issue whether to use minimization techniques in the model theory to implement assumptions, or to use explicit axioms, as in the original situation calculus or with the explanation closure technique. We will discuss each of these issues separately, although they are, of course, closely related.

Our approach is based on making assumptions about event occurrence, both events caused by actions and external events. In this way, it is similar to work by Morgenstern and Stein [1988], Haas [1987], Schubert [1990], and Kowlaski and Sergot [1986]. In such approaches, properties do not change unless there is some event that causes them to change. Approaches based on STRIPS-style representations can be viewed in this way as well, except that the assumptions are not a separable part of the formalism. Rather, the representation only makes sense when these assumptions are made. The other major approach focuses on minimizing property change, with additional constraints based on the temporal ordering of properties (*e.g.*, [Shoham, 1988; Kautz, 1986]) or minimizing causal relationships (*e.g.*, [Lifschitz, 1987]). Sandewall [1994] examines the advantages and limitations of each of these approaches in detail. Most of the approaches that minimize property change cannot handle Sandewall's test suite of problems, let alone be extended to handle external events and simultaneous actions. We find that basing the assumptions on events leads to much more intuitive characterization of problems, where each statement in the logic is closely related to an intuitive fact about the world. In addition, as we show in the subsequent sections, this approach naturally handles a wide range of more complex problems.

The second issue is what mechanism is used to make the assumptions. There are two main approaches here: explicitly adding axioms that encode the assumptions (*e.g.*, [Green, 1969; Schubert, 1990]) or using a nonmonotonic model theory that defines a new notion of entailment that includes the assumptions (*e.g.*, [McCarthy, 1980; Shoham, 1988; Baker, 1991; Sandewall, 1994]). Of course, the work on circumscription shows that model-theoretic techniques always have an equivalent axiomatic formulation, although it may require going beyond standard first-order logic. This equivalence suggests that there is really a continuum of approaches here. Everyone must make assumptions. Some prefer to pack it all in the model theory, others use axioms to capture much of the complexity and use only simple minimization assumptions, while others prefer to encode everything in axioms. Many of the issues come down to ease of formalization, an issue that is bound to vary from researcher to researcher. The reason we prefer explanation closure axioms is that they give us a very flexible system that is easily extended to handle complex issues in representing actions. In addition, the resulting representation is a standard first-order logic, so it is relatively straightforward to tell if certain consequences follow from the axioms. The same cannot be said for approaches based on specialized reasoners or specialized semantic theories that solve one problem at the expense of the others. Good evidence for this claim is seen in the fact that the community has seen the "Dead-Xor-Alive" problem and the Walking Turkey problem as problems at all. They should be trivial consequences of any reasonable theory of action.

Explanation closure axioms allow us to treat each event class on a case-by-case basis. This allows idiosyncratic events and properties to be represented that don't follow the norm. It is much harder to have such flexibility in the model minimization approaches, as all events and properties tend to be treated uniformly. To handle exceptional cases usually requires extending the syntax of the language to introduce special classes of predicates that are minimized in different ways, leading to more complex and unintuitive formulations.

Some common criticisms of explanation closure approaches are that it is too difficult to write down all the axioms, and that the approach is cheating because we have encoded the solution in the axioms. Schubert [1990] provides a good response to the first criticism, namely that the explanation closure axioms are an essential part of an agent's commonsense knowledge of the world, and are needed independently of the frame problem in tasks such as plan recognition or natural language understanding. In addition, just because the theory is based on these axioms doesn't mean that a programmer would have to write them all down. In some cases, the axioms can be automatically computed [Reiter, 1992] or generated on the fly. In an actual application, we can take advantage of such methods whenever possible. As for the issue of cheating, it doesn't seem to us that explicitly encoding the assumptions that make the representation work is any more cheating than hiding the assumptions in a complex model theory. We must remember that these really are assumptions, and they may be wrong. If we are ever to reason about cases where assumptions have been made and shown to be false, they need to be explicit in the representation.

5 External Events

External events arise for different reasons. Some simply occur as a result of natural forces in the world, whereas others are set into motion by the action of the planning agent. Some external events are predictable—we know, for instance, that the sun will rise tomorrow, and that many of the central events of a normal workday will occur. But even when we are certain that a particular event will occur, we are often still uncertain as to when it will occur. Most of the time, for instance, I can only roughly estimate when the sun will rise, but this uncertainty doesn't generally affect my plans. Rather, I know that by a certain time, say 6AM at this time of year, the sun will be up. In most realistic applications, the planning agent will not have complete knowledge of the world and its causal structure, and thus there will be many external events for which it cannot reliably predict whether they will occur or not. Assuming that such events do not occur will lead to highly optimistic plans. Rather, an agent should be able to plan given the uncertainty of external events.

In this paper, we will treat actions by other agents as external events. That is, there is no difference between events that occur because of natural forces and events that occur because of other agent's actions. In a full theory, the main difference between the two classes would result from additional knowledge that is available when reasoning about other agents. In particular, knowledge that other agents are rational will sanction plan recognition inferences, for example.

The logic presented in Section 3 already provides the formalism for representing external events. In fact, the only way to distinguish external events from events caused by the agent's acting is by the presence of an axiom involving the predicate $Try(\pi, t)$, that states that the event was caused by an action attempt. But there is no requirement that all events have axioms relating them to action attempts.

The complications arise in characterizing the appropriate explanation closure axioms. In particular, if one makes the strong event closure assumptions (as in the previous section), that all events are ultimately caused by some action, then external events caused solely by natural forces or other agents cannot be represented. Of course, the explanation closure approach doesn't require such a strong assumption. If fact, we could simply not have closure axioms for some event classes that are caused by natural forces. But this would then allow such events to interfere with the prediction process at every possible opportunity, as you could never prove that the event didn't occur. While this might be the right result in some situations, usually external events are more constrained.

There are two orthogonal aspects of external events that affect the representation. The first relates to uncertainty about whether the event will occur. A non-probabilistic logic can only easily represent the two extremes on this scale: either the event definitely will occur, or it may or may not occur. The second relates to the conditions for an event's occurrence. Again, there is a continuum here between a complete lack of constraints on when the event may occur, and cases where the event will only occur under specific conditions, such as being caused by the agent's actions. Somewhere in the middle of this scale would be events that may only occur within a certain time period, but that are independent of the agent's actions. In this paper, we will consider three common combinations that appear frequently in many domains. These are:

- Triggered events: The external event will not occur unless it is specifically triggered by an agent's actions (*e.g.*, the microwave will heat my coffee only if someone presses the "on" button).

- Definite events: The external event will definitely occur, independent of the agent's actions, although the exact time may be uncertain (*e.g.*, the sun will rise between 5:30 and 6:30AM).

- Spontaneous events: The external event may or may not occur during some time period (*e.g.*, I might win the lottery tonight).

Our representation can also handle more complex cases, such as triggered spontaneous events, which only become possible as a result of some action by the agent.

5.1 Triggered Events

Triggered events are actually already handled by the development so far. In fact, all events in the last section can be viewed as triggered events that characterize the agent's actions. The same approach can be used to handle other triggered external events.

Sandewall's Hiding Turkey Scenario is quite naturally handled as a simple case of reasoning about triggered external events. In this setting, the turkey may or not be deaf. If it is not deaf, then it will go into hiding when the gun is loaded, and thus escape death from the shooting. That is, we are given information about how the turkey will respond to a certain situation, namely that it will hide if it hears the gun being loaded. We must conclude that after the shooting, either the turkey is alive and not deaf, or it is dead.

Many formalizations of this problem treat the turkey hiding as an conditional effect of loading the gun rather than as an external event. If this approach worked for all triggered external events, then it might be justified. But unfortunately, it only works in the simplest of cases, namely when the event caused has no structure or temporal complexity, and so can be characterized as a static effect. For example, consider a situation where the only place the turkey might hide is in a box, and the box has a door that can be closed by the agent. A natural formalization of this would include the following facts:

- If the turkey is not deaf, it will try to hide when the gun is loaded.
- If the box door is open when the turkey tries to hide, it will be hidden in 30 seconds.

You could capture some of this situation by adding a new effect rule about loading that if the box door is open, the turkey will hide. But it is not clear how you might encode the fact that the turkey is not hidden for 30 seconds, so if you shoot quickly you could still hit it. As the scenario becomes more complicated, more and more information would have to be packed into the load action rather than the hiding event. Even if this could be done, it would very unintuitive, as it would not allow you to reason about the turkey hiding except in this one case when the agent loads the gun. What if the turkey may hide for other reasons as well, or might spontaneously decide to hide, or always hides between 6PM and 7PM? Clearly, hiding is just as valid an event as any other event (such as loading), and requires similar treatment.

Given all this, let's return to the simple example as originally formulated. Taking $Deaf$ to be an atemporal predicate, the triggering of hiding by loading is captured with a conditional generation axiom:

EGEN3 $\forall e\,.\,\textsc{Load}(e) \wedge \neg Deaf \supset \exists e'\,.\,\textsc{Hide}(e') \wedge time(e) = time(e')$

To capture the simplicity of the original problem and other solutions in the literature, we assume that the hiding is simultaneous with the loading. The more natural formulation would complicate the proof as we would need to reason about whether the agent could shoot while the turkey was trying to hide.

Next, we need to add axioms for hiding events to the YTS axioms from Figure 7. These are:

EDEF3 $\forall e\,.\,\textsc{Hide}(e) \supset \mathit{Hidden}(\mathit{eff1}(e)) \wedge \mathit{time}(e) : \mathit{eff1}(e)$

EXCP3 $\forall t, t'\,.\,\neg \mathit{Hidden}(t) \wedge \mathit{Hidden}(t') \wedge t : t' \supset \exists e\,.\,\textsc{Hide}(e) \wedge \mathit{time}(e) : t'$

EXCE3 $\forall e\,.\,\textsc{Hide}(e) \supset \neg \mathit{Deaf} \wedge \exists e'\,.\,\textsc{Load}(e') \wedge \mathit{time}(e') = \mathit{time}(e)$

The EDEF3 axiom simply states that a HIDE event results in the turkey being hidden. EXCP3 is a closure axiom that says that things become hidden only as a result of a HIDE event occurring. Closure axiom EXCE3 is justified by the important information implicit in the problem statement that the turkey does not hide unless it hears the gun being loaded.

We then need to modify the definition of shooting (EDEF2) so that it only kills the turkey if it's not hidden:

EDEF2a $\forall e\,.\,\textsc{Shoot}(e) \supset$
$\mathit{Loaded}(\mathit{pre1}(e)) \wedge \mathit{SameEnd}(\mathit{pre1}(e), \mathit{time}(e)) \wedge$
$\neg \mathit{Loaded}(\mathit{eff1}(e)) \wedge \mathit{time}(e) : \mathit{eff1}(e)$

EDEF2b $\forall e\,.\,\textsc{Shoot}(e) \wedge \neg \mathit{Hidden}(\mathit{time}(e)) \supset$
$\neg \mathit{Alive}(\mathit{eff2}(e)) \wedge \mathit{time}(e) : \mathit{eff2}(e)$

Note that this characterization of a SHOOT event means "the bullet leaving the gun," presumably in the direction of the turkey. It does not mean "kill the turkey," which is rather a conditional effect of the shooting (EDEF2b). The action *shoot* is "pulling the trigger" on this account.

The axioms describing the problem are then as follows (the YTS axioms plus new axioms AXH1 and AXH2):

AX0 $t0 \prec t1 \prec t2 \prec t3$

AX1 $\mathit{Alive}(t0)$

AX2 $\neg \mathit{Loaded}(t0)$

AXH1 $\neg \mathit{Hidden}(t0)$

AXH2 $\forall t, t'\,.\,\mathit{Hidden}(t) \wedge t \prec: t' \supset \mathit{Hidden}(t')$

AX3 $\mathit{Try}(\mathit{load}, t1)$

AX4 $\mathit{Try}(\mathit{shoot}, t3)$

AX5 $\forall a, t\,.\,\mathit{Try}(a, t) \equiv (a = \mathit{load} \wedge t = t1) \vee (a = \mathit{shoot} \wedge t = t3)$

Note that axiom AXH2 simply states that once the turkey is hidden it remains hiding forever. A more realistic axiomatization might allow the possibly of an UNHIDE event, and then provide explanation closure axioms for when it might occur. The details are irrelevant for purposes of the example.

To Prove: $\mathit{Deaf} \otimes \exists t\,.\,\mathit{Alive}(t) \wedge t3 \prec: t$

Proof: The following is a sketch of the proof, since many of the details are similar to the examples presented previously.

(a) Suppose $Deaf$. The issue is whether the turkey is hiding at the time of the shooting, *i.e.*, whether $Hidden(t3)$. Suppose it is. Then, from AX0, AXH1, and TRPT we get

$$\exists T, T' \,.\, Hidden(T) \wedge \neg Hidden(T') \wedge T : T' \wedge t3 \sqsubseteq T'.$$

Then EXCP3 gives:

$$\exists e \,.\, \textsc{Hide}(e) \wedge time(e) : T'.$$

But then EXCE3 gives $\neg Deaf$, a contradiction, since the turkey only hides if it is not deaf. Thus $\neg Hidden(t3)$.

As in the YTS, the loading succeeds (ETRY1, EDEF1) and *Loaded* persists from $\mathit{eff1}(e_1)$ through $\mathit{pre1}(e_3)$ (EXCP2a, AX5). Thus the shooting succeeds (ETRY2) and from EDEF2b and the fact that the turkey is not hiding at $t3$ (above), the turkey is dead at $\mathit{eff2}(e_3)$ where $t3 : \mathit{eff2}(e_3)$, which, assuming that reincarnation is ruled out axiomatically, rules out its being alive at any time after $t3$, as required.

(b) Suppose $\neg Deaf$. Then AX3 and ETRY1 give

$$\exists e \,.\, \textsc{Load}(e) \wedge time(e) = t1$$

and then EGEN3 gives

$$\exists e' \,.\, \textsc{Hide}(e') \wedge time(e') = t1.$$

That is, the loading succeeds and the turkey hides, since it is not deaf. This is consistent with EXCE3, and then EDEF3 yields $Hidden(\mathit{eff1}(e')) \wedge t1 : \mathit{eff1}(e')$. This persists indefinitely (AXH2), in particular until $\mathit{pre2}(e_3)$, so the shooting fails to kill the turkey (EDEF2b doesn't apply). Thus *Alive* persists indefinitely starting at $t0$ (EXCP2b, AX5), from which we can derive the desired result. □

Triggered events are very useful for characterizing many domains. For example, whenever you press a button to start a machine, you trigger an event. Our formalism allows you to define the behavior of the machine using the full power of the language to describe events. Thus, pressing one button could trigger a complex sequence of events that greatly affects the domain for extended periods of time. A simple example could involve making a simple meal of reheated pizza. The plan is to put the pizza in the microwave oven and then press the start button. While it is cooking, you take a beer out of the refrigerator and get out the dishes. When the microwave beeps, you take out the pizza and eat. The fact that the microwave is running could constrain other actions you can perform during that time. For instance, you couldn't also reheat some coffee using the microwave, as it is in use. Or if the electric service is limited, you might not be able to run the toaster oven at the same time without blowing a fuse. As simple as this scenario sounds, representing it is beyond the capabilities of most formalisms.

5.2 Definite Events

Definite events can be viewed as a generalization of triggered events, where there are arbitrary constraints on when the events occur. These cases are easily handled in our temporal logic. For instance, if 5AM6 is the time interval between 5AM and 6AM, then the fact that the sun will rise sometime during this time is simply:

$$\exists t, e \,.\, \textsc{SunRise}(t, e) \wedge t \sqsubset 5AM6,$$

where this event is defined by the axiom:

$$\begin{aligned}&\forall t, e \,.\, \textsc{SunRise}(t, e) \supset \\ &\qquad \neg SunShining(pre1(t)) \wedge SunShining(eff1(t)) \wedge pre1(t) : t : eff1(t).\end{aligned}$$

The problem arises as to how the system can infer that the sun doesn't rise (or more importantly set) at other times than the constraints above allow. Given the information as stated, nothing would prevent this. But the problem is not with the formalism. Rather, the problem is that the above axiom doesn't capture all of what we know about the sun rising. In fact, we also know that it rises only once a day. Thus, a revised axiom is needed, and would look as follows, where $5AM6$ is as defined above, and $TODAY$ is the time interval for the entire day, which includes $5AM6$.

$$\begin{aligned}&\exists t, e \,.\, \textsc{SunRise}(t, e) \wedge t \sqsubseteq 5AM6 \wedge \\ &\qquad \forall t', e' \,.\, \textsc{SunRise}(t', e') \wedge t' \sqsubset TODAY \supset e' = e.\end{aligned}$$

In other words, there is a sun rising event sometime between 5 and 6 AM, and this is the only sun rising event today.

Note that by better capturing our actual knowledge of the event, we have constructed a "closure" axiom that gives us exactly the right behavior. It would be a mistake to have tried to handle this case by some uniform strategy over all events (either by model minimization or by automatically generated closure axioms), as the appropriate axiom depends on our specific knowledge about the event. Another external event, say SchoolBusArrives, would have a different characterization as it occurs twice a day, while yet others might have more complex structure.

5.3 Spontaneous Events

The final class of external events to be considered involves events that might occur, in contrast to those the definitely occur as discussed above. Since the agent does not have knowledge of whether events in this class occur or not, it is possible that these events might interfere with the agent's plan any time. As a result, any minimization strategy applied to such events would result in overly optimistic plans based on reasoning similar to proof by failure: if I cannot prove that the event occurs, then it doesn't occur. Of course, this strategy eliminates the possibility of representing spontaneous events.

If an event is always possible, *i.e.*, it might spontaneously occur at any time, then this is represented by the absence of any closure axiom whatsoever. If the event has significant effects, then it will effectively cripple the prediction process. But we would claim that this is exactly right. Consider a turkey-shoot domain again, where another agent may unload the gun at any time. In this situation, it would be impossible to construct a plan that is guaranteed to kill the turkey. This is because at every point in the plan, the other agent might unload the gun before it is fired. Without introducing some notion of probability, this seems the best that a purely logical theory can do. But interesting cases arise when events may occur spontaneously only under certain conditions: say that the other agent only can only unload the gun if the agent is nearby, or that the agent can only interfere in the morning, or that the agent can only unload the gun at most two times. Once there are some constraints on when the spontaneous events might occur, the agent once again may be able to make concrete predictions about the effects of a set of actions.

We need a minor extension to the formalism to allow us to specify constraints on spontaneous events. Specifically, a new predicate *Spontaneous* is introduced that asserts that an event instance spontaneously occured. Events that are never planned, say SNEEZE, are always spontaneous, and this is captured by an axioms of the form

$$\forall e \,.\, \textsc{Sneeze}(e) \supset \mathit{Spontaneous}(e).$$

Events that cannot occur spontaneously are similarly declared to be non-spontaneous, as in the following axiom that says that guns never load spontaneously,

$$\forall e \,.\, \textsc{Load}(e) \supset \neg \mathit{Spontaneous}(e).$$

Events that may occur spontaneously, or may be caused by the agents actions will not have an axiom either way, leaving open both possibilities. More interestingly, events that may be spontaneous under certain conditions can be represented by other axioms. For instance, if the UNLOAD event may occur spontaneously only when the other agent is nearby, the appropriate axiom would be something like:

$$\forall t, e \,.\, \textsc{UnLoad}(t, e) \wedge \neg \mathit{OtherNearBy}(t) \supset \neg \mathit{Spontaneous}(e),$$

i.e., if the other agent is not nearby, then the unload event cannot occur spontaneously. This leaves the possibility open for a spontaneous unloading event under other circumstances. Note that we are encoding a notion of possibility here as a form of ignorance. An event e is spontaneous if you can prove $\mathit{Spontaneous}(e)$, it is not spontaneous if you can prove $\neg \mathit{Spontaneous}(e)$, and it is possibly spontaneous if you can neither prove $\mathit{Spontaneous}(e)$ or $\neg \mathit{Spontaneous}(e)$.

With the possibility of spontaneous events, the closure axioms developed earlier will need to be modified for events that could sometimes occur spontaneously. This is simply accomplished by adding a new disjunct to each one allowing for spontaneous occurrence. For instance, the new closure axiom for UNLOAD would be:

$$\forall t, e \,.\, \textsc{UnLoad}(t, e) \supset \mathit{Try}(\mathit{unload}, t) \vee \mathit{Spontaneous}(e),$$

i.e., an UNLOAD event occurs only if the agent attempts an unload action, or if it is spontaneous. If the other agent isn't nearby, then the previous axiom will eliminate the possibility of an unload event spontaneously occuring, and it would occur only if the agent explicitly unloads it.

As an example, consider the hiding turkey scenario again. Suppose we know that the turkey may always hide spontaneously (*i.e.*, we know no restrictions on when hiding may spontaneously occur). The extended closure axiom would then be:

EXCE3 $\forall e \,.\, \textsc{Hide}(e) \supset$
$[\neg Deaf \wedge \exists e' \,.\, \textsc{Load}(e') \wedge time(e') = time(e)] \vee$
$Spontaneous(e)$

Now, the proof sketched above will not go through. Even if the turkey is deaf, we will not be able to prove that the turkey didn't go into hiding spontaneously. We would still be able to show that if it is not deaf, it will be alive after the shooting, but if it is deaf, we won't know whether it's alive or not.

Of course, the situation could be made more interesting. For instance, maybe we know that the turkey only might spontaneously hide between times $t4$ and $t7$ (say they only hide in the evening). This is easily expressed. The axiom constraining spontaneous hiding events becomes:

$$\forall e \,.\, \textsc{Hide}(e) \wedge (time(e) \prec\colon t4 \vee t7 \prec\colon time(e)) \supset \neg Spontaneous(e).$$

This axiom states that hiding is not spontaneous if it occurs before $t4$ or after $t7$ (and hence allows the possibility of its being spontaneous between $t4$ and $t7$). As a result, the original proof will go through again except for times between $t4$ and $t7$. In particular, we can again prove that the turkey will be killed if it is deaf and we shoot at time $t3$, given appropriate axioms ordering $t0$–$t7$.

It is interesting to observe that if we are only concerned with prediction, we do not need any axioms that explicitly assert when an event is definitely spontaneous, because the prediction process is driven by the closure axioms that require us to prove $\neg Spontaneous(e)$ in order to prove persistence. When considering additional reasoning tasks in the same domain, however, such as explanation, then explicit axioms stating that when an event is spontaneous are very useful. This is because of the abductive nature of explanation, where explanations involving provably spontaneous axioms would require fewer assumptions than explanations that simply assume an event was spontaneous. While we have not considered explanation here, the techniques developed by Kautz [1991] for plan recognition could be adapted easily for use with our representation.

5.4 Discussion

Most formalisms prohibit external events. For example, action definitions in STRIPS-based systems must encode all changes in the world, so external events are fundamentally banned from analysis. Similarly, the constructive situation calculus has the same problem—since there is no way to express information about what the state of the world is while the action is happening, there is no mechanism for allowing an event to occur while an action is being performed. Lifschitz and Rabinov [Lifschitz and Rabinov, 1989] present a limited mechanism to handle this by allowing "miracles" to occur while an action is performed to explain why its effects are not as expected. But this does not allow external events to occur independently of actions, nor does it give events the first-class status required to represent complex situations. In addition, by minimizing the number of miracles, the approach makes it difficult to handle uncertainty about whether some external event will occur or not. Another possibility would be to encode external events as pseudo-actions so that they can be used in the result function, although they are clearly different from normal actions (*e.g.*, the agent couldn't plan to execute them).

In addition, the situation calculus typically only represents possible futures from the initial state, and has no notion of an actual future. Thus, one cannot simply state that an event E will occur, say at noon tomorrow. The obvious way to capture this is by an axiom that guarantees that every possible action sequence that extends past noon tomorrow includes E at the appropriate place in the sequence. To represent such information, you need to be able to quantify over all possible action sequences, including those where actions overlap. While the formalism can be extended in this way, one loses the constructivity that made the constructive situation calculus an attractive formalism for reasoning about action.

Even more problematic are external events that are conditional on the agent's behavior, or simply may or may not occur independent of the agent. If some event E occurs, then the reasoner must only consider situations that are constructed by an action sequence with E in it and if it doesn't occur, then E should not be in the sequence. But this forces a change in the interpretation of situations. Without external events, every situation (defined by an action sequence) indicated a possible attainable future, and to find a plan for a goal G, one needed only to find a sequence that resulting in a situation where G was true. With external events, there may be sequences that produce a situation in which G is true, but finding one does not guarantee that the goal can be achieved, since the situation may not be attainable. For example, a situation in which G is true might be constructed using an external event E, but the agent cannot guarantee that E will occur, so this situation may not be attainable.

Work based on the situation calculus that addresses external events typically rejects the constraints of the constructive approach and uses the more general formalism. In these approaches situations are arbitrary states of the world, not necessarily related to a particular action sequence. In addition, situations can be associated with times from a time line and one can quantify over situations. With this, one can introduce an $Occurs$ predicate that asserts that a particular action occurs at a specific time and that is defined by an axiom that quantifies over all situations at that time (*e.g.*, [Pinto, 1994; Miller and Shanahan, 1994]). This development moves the formalism closer to what we are proposing.

Ultimately, the main difference between our approach and these extended situation calculus representations with explicit time will probably be one of approach. We start from a representation of the actual world (or an agent's beliefs about the world), and must introduce mechanisms to allow reasoning about the effects of possible actions. The situation calculus starts from a representation based on possible futures based on the actions, and must introduce mechanisms for dealing with information about the actual world. While reasoning about external events is going to be complicated in any formalism, we believe that our approach is superior for several reasons. First, it allows problems to be formalized in a simpler and, we feel, more direct manner. Second, our representation is closer to the types of representations used in implemented knowledge representation systems, and thus appears more suitable for deriving practical applications, such as planners and natural language understanding systems. The main weakness is the lack of a notion of possible futures. But a modal operator for possibility is needed independently of temporal reasoning and reasoning about action. Combining such a modal operator with our linear time model

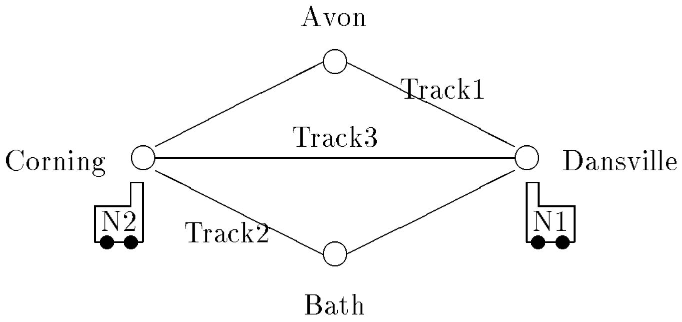

Figure 10: Example TRAINS domain map

would allow us to express anything that is expressible in the situation calculus. There does not seem to be any advantage to combining the two needs in a single mechanism. Interestingly, this paper demonstrates that a wide range of problems can be solved without ever using the explicit modality.

6 Simultaneous Action

There are several levels of difficulty in dealing with simultaneous actions. The model described so far already handles some cases of simultaneous actions, namely

- When two actions cannot occur simultaneously;
- When they occur simultaneously and are independent of each other; and
- When they together have additional effects that neither one would have individually.

The more difficult case is when two actions partially or conditionally interfere. After first describing how the basic approach handles the three cases above, we will discuss some approaches to reasoning about interference.

We prefer to use a more complex domain than that of the YTS to illustrate these cases. The TRAINS domain is a transportation and manufacturing domain developed for use in the TRAINS project [Allen and Schubert, 1991]. The goal of the project is to build an intelligent planning assistant that is conversationally proficient in natural language. The domain involves several cities connected by rail links, warehouses and factories at various cities, and engines, boxcars, *etc.*., to transport goods between them. A simple TRAINS domain map used in the examples to follow is shown in Figure 10. In this domain, interacting simultaneous actions and external events are unavoidable aspects of the domain.

6.1 Independent and Mutually-Exclusive Actions

When actions are independent of each other, the existing formalism does exactly the right thing. The effect of the two actions is simply the sum of the effects of actions individually, a result you get directly from the axioms. When there are simple interactions, such as mutual exclusion of actions under certain circumstances, this behavior typically naturally falls out of a reasonable axiomatization of the domain.

For example, consider a simplified axiomatization concerning engines moving between two cities, where n ranges over engines, $c1$ and $c2$ over cities, and x over tracks, and $move(n, c1, c2, x)$ is an action that moves engine n from $c1$ to $c2$ along track x:

ETRY1 $\forall n, c1, c2, x, t \,.\, Try(move(n, c1, c2, x), t) \supset On(n, x, t)$

ETRY2 $\forall n, c1, c2, x, t \,.\, Try(move(n, c1, c2, x), t) \wedge TrackClearFor(n, x, t) \supset$
$\exists e \,.\, \textsc{MoveBetween}(n, c1, c2, x, t, e)$

EDEF2 $\forall n, c1, c2, x, t, e \,.\, \textsc{MoveBetween}(n, c1, c2, x, t, e) \supset$
$At(n, c2, \mathit{eff1}(e)) \wedge t : \mathit{eff1}(e)$

ETRY1 states that while trying to move along the track x from city $c1$ to city $c2$, the engine n is on the track. ETRY2 states that if the track is clear when the agent attempts to move the engine, then a MoveBetween event occurs. Finally, EDEF2 states that if a MoveBetween occurs, then the engine ends up at the destination city.

In addition to these axioms, we have general knowledge about the domain that includes the fact that an object cannot be in two different places at once and what is means for the track to be clear for an engine:

AX1 $\forall n, x, y, t, t' \,.\, On(n, x, t) \wedge On(n, y, t') \wedge x \neq y \supset t \bowtie t'$.

AX2 $\forall n, x, t \,.\, TrackClearFor(n, x, t) \equiv$
$\forall n', t' \,.\, (t' \sqsubseteq t \wedge Engine(n')) \supset \neg On(n', x, t') \vee n' = n$

Note that we have not added any special knowledge about how the *move* actions interact, but these axioms already capture a good range of our intuitions about the domain. For example, consider the scenario shown in Figure 10 with two engines, $N1$ at Dansville and $N2$ at Corning. If $N1$ tries to move to Avon on Track 1, and $N2$ tries to move to Bath on Track 2 simultaneously, then they both get to their destinations. The scenario is axiomatized as follows:

AX3 $t0 : t1$

AX4 $Engine(N1) \wedge Engine(N2) \wedge \forall n \,.\, Engine(n) \supset (n = N1) \vee (n = N2)$

AX5 $Track1 \neq Track2$

AX6 $At(N1, Dansville, t0)$

AX7 $At(N2, Corning, t0)$

AX8 $Try(move(N1, Dansville, Avon, Track1), t1)$

AX9 $Try(move(N2, Corning, Bath, Track2), t1)$

We can prove that there is a time t, where $t1 : t$, and where both $At(N1, Avon, t)$ and $At(N2, Bath, t)$ are true.

Proof: Consider proving $At(N1, Avon, t)$. From ETRY1 and AX9 we get

$$On(N2, Track2, t1),$$

then from AX1 we get $\neg On(N2, Track1, t1)$. AX2 and AX4 then give

$$TrackClearFor(N1, Track1, t1),$$

which together with AX8 and ETRY2 gives

$$\exists e_1 \,.\, \textsc{MoveBetween}(N1, Dansville, Avon, Track1, t1, e_1).$$

Then EDEF2 gives $At(N1, Avon, \mathit{eff1}(e_1))$ and $t1 : \mathit{eff1}(e_1)$, as required. An exactly parallel proof shows that $At(N2, Bath, \mathit{eff1}(e_2))$ and $t1 : \mathit{eff1}(e_2)$, so we have shown that both actions had their intended effects. □

When these actions interfere with each other, the domain knowledge will block the unwanted inference that the events occured successfully together. For instance, consider a variant of the above scenario where both engines try to use Track 3, *i.e.*, we have:

AX8 $Try(move(N1, Dansville, Corning, Track3), t1)$

AX9 $Try(move(N2, Corning, Dansville, Track3), t1)$

In this case, we will not be able to prove that either MoveBetween event occurs. The reason is that the two move actions create a situation in which both engines are on the same track. This then prevents ETRY2 from being used, and so we can conclude very little about what events were caused by these actions.

Notice that the fact that the actions were attempted is still true. In general, we do not limit action attempts in any way, reflecting a belief that an agent may attempt most actions at any time. If the conditions are not appropriate, the actions simply won't cause the desired events. Of course, we could add additional axioms describing what happens when action attempts interact. For instance, we could add a dramatic axiom that asserts that if two move actions are attempted simultaneously on the same track, the engines crash. More realistically, we might predict a deadlock situation, or events in which the engines end up returning to their originating cities. The point here is that the temporal logic provides a fairly natural way to axiomatize the knowledge in this domain so as to capture the desired behavior.

Another form of interaction between simultaneous action attempts are resource conflicts. For instance, in the TRAINS domain, an engine requires fuel to make trips. If two engines are planning trips but there is only enough fuel at the station for one, then only one engine can succeed. Many of these cases require reasoning about quantities, which would introduce many complications not necessary to make the points in this paper. But some resources can be modeled as unitary—either you have the resource or you don't. In these cases, the techniques used above can be adapted to resources. In fact, you can view the track in the previous example as a resource required by an engine to make a trip, and the problem where the engines try to use the same track as a resource conflict.

Other constraints on simultaneous action arise because of limitations on the part of the agent. While often such restrictions can be cast in terms of resource conflicts, the temporal logic also allows specific axioms about action co-occurrence. For instance, say a engineer cannot try to couple one car and decouple another at the same time. This constraint can be simply captured by the axiom:

$$\forall n, c, c', t, t' \,.\, Try(couple(n, c), t) \wedge Try(decouple(n, c'), t') \supset t \bowtie t'$$

This axiom would make a set of assertions where an agent simultaneously attempted to couple one car and decouple another inconsistent. Axioms like this can be very useful in defining simple planning systems similar to non-linear planners (*e.g.*, [Allen and Koomen, 1983]).

6.2 Synergistic Effects

It is also easy to define additional synergistic effects that occur when two actions occur simultaneously. For instance, you can define events that occur only when several actions are performed simultaneously. Thus, if one agent lifts one end of a piano while another agent lifts the other end, then the entire piano is lifted off the floor. Such situations are easily described in our representation. For example, assuming axioms for a $lift$ action that causes LIFT events, we could specify how the simultaneous lifting of both ends of the piano results in the piano being lifted using a generation axiom such as:

$$\forall p, t, e1, e2 \,.\, \textsc{Lift}(left(p), t, e1) \wedge \textsc{Lift}(right(p), t, e2) \supset \exists e3 \,.\, \textsc{Lift}(p, t, e3).$$

The fact that actions and events have durations allows us to make this somewhat more realistic by only requiring that the lifting of the ends overlap, rather than be simultaneous, as in

$$\begin{aligned} &\forall p, t1, t2, e1, e2 \,.\, \textsc{Lift}(left(p), t1, e1) \wedge \textsc{Lift}(right(p), t2, e2) \wedge \neg(t1 \bowtie t2) \supset \\ &\qquad \exists e3 \,.\, \textsc{Lift}(p, t1 \cap t2, e3). \end{aligned}$$

The function "$t1 \cap t2$" refers to the interval "common" to both $t1$ and $t2$, which must be non-disjoint (as they are here). An even more realistic version of the axiom would require that the ends were both lifted for long enough.

A more realistic example of synergistic effects involves performing an action while other events are occurring (possibly also caused by actions), to cause additional effects. Consider an example from the TRAINS world that illustrates this. To decouple a car, an engineer must activate the decoupler while the engine is moving forward. The actions of the engineer are not simultaneous here, as the engine moving forward is an event triggered by the engineer's action of setting the throttle. This scenario can be axiomatized as follows:

ETRY1 $\forall n, t \,.\, Try(setThrottle(n), t) \supset \exists e, t' \,.\, \textsc{Move}(n, t', e) \wedge t : t'$

ETRY2 $\forall n, t \,.\, Try(activate(n), t, e) \supset \exists e \,.\, \textsc{Activate}(n, t, e)$

ETRY1 states that setting the throttle triggers the moving event. ETRY2 states that the decoupler is activated only while the engineer holds down the activator button.

These two events need to occur simultaneously to uncouple a car—neither one alone results in the car being uncoupled. This knowledge is captured by a conditional generation rule that involves two events on the left-hand side. Here's a possible axiomatization:

EGEN3 $\forall n, c, x, t1, t2, e1, e2, t \,.\, \textsc{Move}(n, t1, e1) \wedge \textsc{Activate}(n, t2, e2) \wedge$
$\neg(t1 \bowtie t2) \wedge \mathit{Coupled}(n, c, t) \wedge t : (t1 \cap t2) \supset$
$\exists e \,.\, \textsc{Uncouple}(n, c, (t1 \cap t2), e)$

EDEF3 $\forall n, c, t, e \,.\, \textsc{Uncouple}(n, c, t, e) \supset \neg \mathit{Coupled}(n, c, \mathit{eff1}(e)) \wedge t : \mathit{eff1}(e)$

What EGEN3 says is that if the Move and Activate events overlap temporally, and if the engine and car are coupled prior to the simultaneous moving and activating, then an Uncouple event will occur (over the time during which the events overlap). EDEF3 simply states that an effect of uncoupling is that the engine and car are not coupled.

To complete this example, we would need to encode some commonsense closure axioms, such as the fact that the Move event continues until the engineer unsets the throttle. With these, we could prove that if the engineer sets the throttle and then activates the decoupler, and does nothing else, then the simultaneous occurrence of the Move and Activate events will result in the car being decoupled.

6.3 Interference

Complications in handling simultaneous actions and events arise when they partially interfere with each other, or interfere under certain conditions. Note that with our view of events as classifications of patterns of change, it makes no sense to talk about events interfering with each other. The same world cannot be characterized by two events whose definitions are mutually inconsistent. Interference makes sense only when talking about actions. For instance, if an agent is trying to open a door, then it is performing a *push* action hoping to cause an OpenDoor event. If, simultaneously, another agent pushes on the door the other way, then the first agent will still have performed the *push* action, but the OpenDoor event will not have occurred. Thus, we say that the two *push* actions interfered with each other. In this case, they cancelled each other out. Of course, action attempts might also be affected by external events that are occurring simultaneously. We will consider both cases below.

When an action interacts with other events that are occurring, the interaction can be treated much the same as we handled conditional effects of action attempts when different properties hold. This technique formalizes the problem in terms of events, and the different effects result from synergistic interaction between events. For example, consider a situation where we want to load a boxcar from an automatic hopper. If we push the start button, the hopper tips and dumps its cargo into the boxcar. This, of course, requires that the boxcar be in the appropriate location under the hopper. If the boxcar was not under the hopper to start with, it is clear that the loading of the car would not occur although the hopper would still be emptied. The more interesting case occurs when the boxcar is initially under the hopper, but then moves away while being loaded. Intuitively, we say that the moving of the boxcar interferes with the loading of the car. Again, the hopper would still end up empty, but the boxcar would not contain all the cargo. Here's a simple axiomatization of these actions and events.

First, pushing the start button causes an EMPTYHOPPER event to occur, and occurrence of an EMPTYHOPPER event entails that the hopper is empty afterwards:

ETRY1 $\forall h, t \,.\, Try(startHopper(h), t) \supset \exists e, t' \,.\, \textsc{EmptyHopper}(h, t', e) \wedge t : t'$

EDEF1 $\forall h, t, e \,.\, \textsc{EmptyHopper}(h, t, e) \supset Empty(h, \mathit{eff1}(e)) \wedge t : \mathit{eff1}(e)$

In addition, we need another axiom about moving that captures fact that when a MOVE event is occurring, the car is continually changing position:

AX1 $\forall c, t, e, t1, t2 \,.\, \textsc{Move}(c, t, e) \wedge (t1 \sqsubset t) \wedge (t2 \sqsubset t) \wedge (t1 \prec\!: t2) \supset$
$\exists l1, l2 \,.\, l1 \neq l2 \wedge At(c, l1, t1) \wedge At(c, l2, t2)$

And, of course, you cannot be two place at the same time:

AX2 $\forall o, l, l', t, t' \,.\, At(o, l, t) \wedge At(o, l, t') \wedge l \neq l' \supset t \bowtie t'$.

Finally, the information that the boxcar must be under the hopper to be loaded is cast as an event generation axiom, followed by an axiom defining loading events:

EGEN2 $\forall c, h, t, e \,.\, EmptyHopper(h, t, e) \wedge At(c, h, t) \supset \exists e' \,.\, Load(c, h, t, e')$

EDEF2 $\forall c, h, t, e \,.\, Load(c, h, t, e) \supset Loaded(c, \mathit{eff1}(e)) \wedge t : \mathit{eff1}(e)$

We think this is an uncontroversial characterization of some of the general knowledge an agent would need to reason about this domain. And this is all that is needed to reason about the interaction. For instance, consider a situation in which the agent pushes the start button when the boxcar is under the hopper, but the boxcar is moving. Under suitable assumptions about the duration of events, the definition of the MOVE event can be used to prove that there is a time during the EMPTYHOPPER event when the boxcar is not under the hopper. Thus, a LOAD event cannot be inferred from axiom EGEN2. Of course, this is a crude axiomatization for the purposes of illustrating the technique. A better characterization would have axioms that enable reasoning about the changes that occur during the loading action. While this can be done, it doesn't add to the point being made here.

It may be true that we could characterize all interactions between simultaneous actions in terms of events in this way. Ultimately, however, this would require reducing everything to events characterizing physical force equations. In fact, this is what is suggested in many papers addressing simultaneous action (*e.g.*, [Pednault, 1986]). We do not believe this is appropriate for a commonsense theory of action, as it requires agents to understand the physics of the world at a fairly deep level in order to understand simple everyday interactions.

Good examples that show a need for another method can be found in characterizing an agent's knowledge about how its own action attempts interact when performed simultaneously. For instance, a two-armed robot might know that (1) it can lift either a large or a small block individually, (2) it can lift two small blocks simultaneously, but (3) if it tries to lift two large blocks, it will tip over. This knowledge is captured directly in the following axioms:

ETRY1 $\forall b, t \,.\, Block(b) \wedge Try(lift(b), t) \wedge$
$\quad\neg\exists b', t' \,.\, Overlaps(t', t) \wedge Try(lift(b'), t') \supset \exists e \,.\, \textsc{Lift}(b, t, e)$

ETRY2 $\forall b1, b2, t \,.\, SmallBlock(b1) \wedge SmallBlock(b2) \wedge$
$\quad Try(lift(b1), t) \wedge Try(lift(b2), t) \supset$
$\quad\quad \exists e1, e2 \,.\, \textsc{Lift}(b1, t, e1) \wedge \textsc{Lift}(b2, t, e2)$

ETRY3 $\forall b1, b2, t \,.\, LargeBlock(b1) \wedge LargeBlock(b2) \wedge$
$\quad Try(lift(b1), t) \wedge Try(lift(b2), t) \supset \exists e \,.\, \textsc{TipOver}(t, e)$

AX1 $\forall e, e', t, t' \,.\, \textsc{Lift}(t, e) \wedge \textsc{TipOver}(t', e') \supset t \bowtie t'$

Now, some will criticize this solution, saying we just wrote out the answer. But this is exactly the knowledge that an agent would acquire after a short amount of experimentation in the domain. Also, it is reasonable to assume that the agent has relatively complete knowledge of what actions it will attempt over a certain period of time, so negative conditions such as the one used in ETRY1 would be easy to prove. In fact, Haas [1992] has used this idea to build a reactive planner that uses explanation closure. In any event, a formalization at this level seems more more plausible than a solution that solves this problem by reasoning about force equations, which would require much more detailed knowledge of the domain.

We could also use something like McCarthy's Ab predicate [McCarthy, 1980] in ETRY1, which would allow us to incrementally add conditions that entail an abnormal lifting attempt. However, if we used a standard nonmonotonic method to minimize Ab, we would risk losing information about cases where the robot is uncertain. For instance, the axioms above say nothing about what would happen if the agent simultaneously lifts a small block and a large block. This reflects the fact that the agent doesn't know. With a technique based on predicate minimization (such as [Lin and Shoham, 1992]), the agent would assume that it would not tip over, possibly a dangerous assumption. We would instead use explanation closure techniques on the Ab predicate which allow us to leave such cases uncertain.

The agent might also have knowledge about how its actions interact with external events. For example, consider an agent's knowledge about driving a car. The action of turning on the ignition starts the event of the engine running:

$$\forall c, t \,.\, Try(turnIgnition(c), t) \wedge level(c, t) > 0 \supset$$
$$\exists e, t' \,.\, \textsc{RunEngine}(c, t', e) \wedge t : t',$$

where the function $level(c, t)$ produces the minimum level of the gas during time t. When the engine is running, pressing the accelerator causes the car to move forward:

$$\forall c, t \,.\, Try(pushAccelerator(c), t) \wedge$$
$$(\exists e, t' \,.\, \textsc{RunEngine}(c, t', e) \wedge t \sqsubseteq t') \supset \exists e' \,.\, \textsc{Move}(c, t, e').$$

These two axioms capture the interaction between the agent's action and a simultaneously occurring external event directly without requiring a deep understanding of the underlying causal model of how cars work. In addition, the agent might have other knowledge about the engine running event, such as the fact that it requires gasoline in the tank, and consumes gasoline as it runs:

$$\forall c, t, e \,.\, \textsc{RunEngine}(c,t,e) \supset level(c,t) > 0 \wedge$$
$$\forall t1, t2 \,.\, (t1 \sqsubset t) \wedge (t2 \sqsubset t) \wedge (t1 \prec\!: t2) \supset level(c,t1) > level(c,t2).$$

Note that this last axiom could not be expressed if the RunEngine event were formalized as a static property that is a precondition for the moving the car. Thus, to capture all this information in a single framework, the engine running should be formalized by an event.

6.4 Discussion

For the most part, treatments of simultaneous actions in the literature are limited. As mentioned earlier, STRIPS-based systems (*e.g.*, [Tate, 1977; Vere, 1983; Wilkins, 1988]) only allow simultaneity when the actions are independent. A few others (*e.g.*, [Pednault, 1986]) allow for synergistic effects cast in terms of domain constraints on states (*e.g.*, in any state s, if the left side of the piano is lifted, and the right side is lifted, then the piano is lifted).

The situation calculus shows more promise with the introduction of action composition operators. For instance, given two actions a_1 and a_2, then the action $a_1 + a_2$ is the action of performing the two simultaneously (*e.g.*, [Gelfond *et al.*, 1991; Schubert, 1990]). Explicit axioms can then be given for these complex actions, and mechanisms can be introduced to automatically derive such axioms from the individual actions if they are independent of each other (*e.g.*, [Lin and Shoham, 1992]). If the actions are not independent of each other, some reasonable solutions can be found and, as long as actions are instantaneous, it appears that the theory can remain constructive. But these approaches do not seem to be easily extended to handle the more complex cases in which actions have duration and may be temporally related in complex ways. Pelavin [1991] contains an extensive analysis of the problems that arise in general with simultaneous interacting actions.

Some work has explored the possibility of allowing a duration for each action [Gelfond *et al.*, 1991]. Then the duration of a sequence of actions would simply be the sum of the individual action durations. But there is little point to doing this unless one allows these extended actions to overlap with other actions and external events. If these cases are not allowed, adding the durations adds little additional expressive power as the problems that can be stated with durations are effectively isomorphic to problems without the durations. But it is not clear how overlapping and interacting actions can be handled. For instance, if a temporally extended action A maps a situation s to a situation $A(s)$ after a delay of n seconds, how would this combine with another action B that starts 3 seconds after A begins, where B maps a situation s to $B(s)$? If the axioms for A are not conditional, then we end up in state $A(s)$ no matter whether action B occurs or not. If the axioms for A depend on whether B occurs or not, what is the final situation? Clearly, $A(B(s))$ is not right, as it implies that B was executed first and A second, and $B(A(s))$ makes similarly bad claims the other way. Of course, since the situation calculus is embedded in first-order predicate calculus, and we have introduced time into the representation, we should be able to express some axioms that would characterize the what happens when B occurs 3 seconds after the start of A. But by the time we introduce all the temporal mechanisms, it is not clear whether the mechanism of the original situation calculus helps matters, or just gets in the way.

Our approach provides a range of techniques for representing information about interacting actions. In cases where one has detailed knowledge of the causal structure of a domain, the problems can usually be reduced to a level of independent events, and knowledge about interactions are captured by event generation axioms. In other cases, it is more natural to have explicit knowledge about how certain actions interact with events. We believe this flexibility is important for building comprehensive knowledge bases that contain information about a wide range of situations and that are applicable for different reasoning tasks. In addition, we have shown that we can handle these complex cases without introducing any new mechanisms beyond the basic logic discussed in Section 3.

7 Problems and Future Work

One of the most significant open issues in this paper has to do with temporal durations. A realistic characterization of almost all the examples in this paper would require such a capability. For instance, it is not realistic to say that if an agent tries to turn the ignition on the car for any length of time, then the engine will start. If the action is tried for too short a time, the engine probably won't catch. And if the action is tried for too long a time, the starting motor will burn out. So durations play a critical role. At first glance, adding durations does not pose a problem. One simply defines a function that, given an interval, returns a value on some metric scale. A small number of axioms are required to define the appropriate properties of this function. We have not pursued this here as it would further complicate the examples and remove attention from our main points, and in any case, this simple metric model doesn't solve the problem. Consider the action of starting the engine again. There is no minimum time or maximum time within which the engine is guaranteed to start. The duration required depends on the condition of the car, the weather, and many other factors that the agent simply won't have access to. A better formalization of the durational constraints would be that the agent turns the ignition until the engine starts, or a certain time elapses and the agent gives up for fear of burning out the motor, or the battery runs flat. The logic we propose offers no direct solution to this problem, and it remains an important challenge.

Another important issue is the introduction of probabilistic knowledge. By staying within standard first order logic, for example, we are restricted to saying that an event will definitely occur, or that it might possibly occur. We cannot say that an event is very likely to occur, or that it is very unlikely to occur. Such knowledge is crucial for predicting the likely effects of actions and thus for evaluating the likelihood of success for proposed plans. Further, since all formalisms must make assumptions, the ultimate evaluation should be based on how likely the assumptions are to hold. This is another motivation for favoring the explanation closure technique. It makes the assumptions that are made explicit, and thus potentially available for probabilistic analysis. Some initial work on this is described in [Martin, 1993; Martin and Allen, 1993]. It is much more difficult to see how techniques that build the assumptions into the semantic model could be extended to support probabilistic reasoning.

Finally, it needs to be acknowledged that formalizing knowledge using the more expressive temporal representation can be difficult. Subtle differences in meaning and interactions between axioms may be more common than in less powerful representations, and more experimentation is needed in building knowledge bases based on our representation. But it seems to us that this is the price to pay if we want to move beyond Yale Shooting and the Blocks World. Our representation is based on intuitions about the way people describe and reason about actions in language, which we believe makes it more natural, intuitive, and grounded in common sense.

8 Conclusion

Many of the techniques we have proposed have appeared in various forms previously in the literature. What makes this work novel is the combination of the techniques into a unified, and what we think is an intuitive and relatively simple framework. This logic has the following features:

- It can express complex temporal relationships because of its underlying temporal logic.
- It supports explicit reasoning about action attempts and the events they cause, which allows explicit reasoning about success and failure of attempted actions, and subsumes work on conditional effects.
- It handles external events in a natural way, making only minimal distinctions between external events and events caused by the acting agent.
- It handles simultaneous actions in a direct manner, including cases of simultaneously interacting actions.

By using an explicit temporal logic with events, we have been able to handle all these problems within a standard first-order logic. We believe this formalism to be more powerful than competing approaches. We also believe that it is simpler than these other formalisms will be once they are extended to handle the temporal complexities of realistic domains, assuming they can be successfully extended.

The second contribution of this paper has been to add to the argument that explicit frame axioms, based on the explanation closure approach, produce a viable theory of prediction with some attractive formal and practical properties. On the formal side, this approach does not require the introduction of specialized syntactic constructs into the language or the use of nonmonotonic semantic theories. On the practical side, the closure axioms capture knowledge that is required in any case as part of the specification of the domain. Note that the formulation of the temporal logic and the use of explanation closure are separable issues. We would not be surprised if the nonmonotonic techniques in the literature could be adapted to our temporal logic. Whether this could be done without extending the syntax with special operators specifically introduced to enable the right minimizations is more questionable, however. The risk is that such operators might subtly affect the range of situations that can be handled, and may require non-intuitive formulations of problems.

While this paper has focussed on formal issues, it is important to remember that our goal is actually the development of practical planning and natural language understanding systems. As a result, another key requirement on this model is that it provides insight and guidance in developing effective knowledge representation systems. We are using this formalism in the TRAINS project [Allen and Schubert, 1991; Traum *et al.*, 1994; Ferguson and Allen, 1994] and have found that it allows us to express the content of natural language utterances quite directly, and supports plan reasoning algorithms similar to those in the planning literature. The use of explicit assumptions and closure reasoning is essential in such an interactive system where plans are formed based on assumptions and where those assumptions are often the subject of the conversation.

References

[Allen, 1983] James F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, 26(11):832–843, 1983. Also in *Readings in Knowledge Representation*, R.J. Brachman and H.J. Levesque (eds.), Morgan Kaufmann, 1985, pp. 509–522.

[Allen, 1984] James F. Allen, "Towards a general theory of action and time," *Artificial Intelligence*, 23:123–154, 1984. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 464–479.

[Allen, 1991] James F. Allen, "Temporal Reasoning and Planning," in *Reasoning about Plans*, pages 1–68. Morgan Kaufmann, San Mateo, CA, 1991.

[Allen, 1994] James F. Allen, *Natural Language Understanding, 2nd ed.*, Benjamin/Cummings Publishing Co., Menlo Park, CA, 1994.

[Allen and Hayes, 1989] James F. Allen and Patrick J. Hayes, "Moments and points in an interval-based temporal logic," *Computational Intelligence*, 5(4):225–238, 1989.

[Allen and Koomen, 1983] James F. Allen and Johannes A. Koomen, "Planning using a temporal world model," in Alan Bundy, editor, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 741–747, Karlsruhe, West Germany, 8–12 August 1983. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 559–565.

[Allen and Schubert, 1991] James F. Allen and Lenhart K. Schubert, "The TRAINS Project," TRAINS Technical Note 91-1, Department of Computer Science, University of Rochester, Rochester, NY, May 1991.

[Bacchus *et al.*, 1989] Fahiem Bacchus, Josh Tenenberg, and Johannes A. Koomen, "A non-reified temporal logic," in R.J. Brachman, H.J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, pages 2–10, Toronto, Ont., 15–18 May 1989.

[Baker, 1991] Andrew Baker, "Nonmonotonic reasoning in the framework of the situation calculus," *Artificial Intelligence*, 49:5–24, 1991.

[Brown, 1987] Frank M. Brown, editor, *Proceedings of the 1987 Workshop: The Frame Problem in Artificial Intelligence*. Lawrence, KA, Morgan Kaufmann, 12–15 April 1987.

[Chapman, 1987] David Chapman, "Planning for conjunctive goals," *Artificial Intelligence*, 32:333–377, 1987. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 537–558.

[Davidson, 1967] Donald Davidson, "The logical form of action sentences," in N. Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press, 1967. Excerpted in *The Logic of Grammar*, D. Davison and G. Harmon (eds.), Dickenson Publishing Co., 1975, pp. 235–245.

[Davis, 1992] Ernest Davis, "Infinite loops in finite time: Some observations," in Bernard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR92)*, pages 47–58, Boston, MA, 25–29 October 1992. Morgan Kaufmann.

[Dean and McDermott, 1987] Tom Dean and Drew McDermott, "Temporal data base management," *Artificial Intelligence*, 32:1–55, 1987. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 596–623.

[Dowty, 1986] David R. Dowty, "The Effects of Aspectual Class on the Temporal Structure of Discourse: Semantics or Pragmatics?," *Linguistics and Philosophy*, 9(1), 1986.

[Eshghi, 1988] Kave Eshghi, "Abductive planning with event calculus," in R.A. Kowalski and K.A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 562–579, Cambridge, MA, 1988. MIT Press.

[Ferguson, 1992] George Ferguson, "Explicit Representation of Events, Actions, and Plans for Assumption-Based Plan Reasoning," Technical Report 428, Department of Computer Science, University of Rochester, Rochester, NY, June 1992.

[Ferguson and Allen, 1993] George Ferguson and James F. Allen, "Generic Plan Recognition for Dialogue Systems," in *Proceedings of the ARPA Workshop on Human Language Technology*, Princeton, NJ, 21–23 March 1993.

[Ferguson and Allen, 1994] George Ferguson and James F. Allen, "Arguing about Plans: Plan Representation and Reasoning for Mixed-Initiative Planning," in *Proceedings of the Second International Conference on AI Planning Systems (AIPS-94)*, Chicago, IL, 15–17 June 1994.

[Fikes and Nilsson, 1971] Richard E. Fikes and N.J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, 2:198–208, 1971. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 88–97.

[Gelfond *et al.*, 1991] Michael Gelfond, Vladimir Lifschitz, and Arkady Rabinov, "What are the limitations of the situation calculus?," in *Proceedings of the AAAI Symposium on Logical Formalizations of Commonsense Reasoning.* Stanford University, 26–28 March 1991.

[Georgeff, 1986a] Michael P. Georgeff, "Actions, Processes, and Causality," in Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Los Altos, CA, 30 June–2 July 1986. Morgan Kaufmann.

[Georgeff, 1986b] Michael P. Georgeff, "The representation of events in multiagent domains," in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 70–75, Philadelphia, PA, 11–15 August 1986. University of Pennsylvania.

[Goldman, 1970] Alvin I. Goldman, *A Theory of Human Action*, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[Green, 1969] Cordell Green, "An application of theorem proving to problem solving," in Donald E. Walker, editor, *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69)*, pages 741–747, Washington, DC, 7–9 May 1969. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 67–87.

[Haas, 1992] Andrew R. Haas, "A reactive planner that uses explanation closure," in Bernard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR92)*, pages 93–102, Boston, MA, 25–29 October 1992. Morgan Kaufmann.

[Haas, 1987] Andrew W. Haas, "The case for domain-specific frame axioms," in Frank M. Brown, editor, *Proceedings of the 1987 Workshop: The Frame Problem in Artificial Intelligence*, pages 343–348. Lawrence, KA, Morgan Kaufmann, 12–15 April 1987.

[Hamblin, 1972] C. L. Hamblin, "Instants and Intervals," in J. T. Fraser, F. C. Haber, and G. H. Müller, editors, *The Study of Time*, pages 324–328. Springer-Verlag, New York, 1972.

[Hanks and McDermott, 1986] S. Hanks and D. McDermott, "Default reasoning, nonmonotonic logic, and the frame problem," in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 328–333, Philadelphia, PA, 11–15 August 1986. University of Pennsylvania.

[Hobbs *et al.*, 1993] Jerry R. Hobbs, Mark E. Stickel, Douglas E. Appelt, and Paul Martin, "Interpretation as abduction," *Artificial Intelligence*, 63:69–142, 1993.

[Kautz, 1986] Henry A. Kautz, "The logic of persistence," in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 401–405, Philadelphia, PA, 11–15 August 1986. University of Pennsylvania.

[Kautz, 1991] Henry A. Kautz, "A Formal Theory of Plan Recognition and its Implementation," in *Reasoning about Plans*, pages 69–126. Morgan Kaufmann, San Mateo, CA, 1991.

[Kowalski, 1992] Robert Kowalski, "Database updates in the event calculus," *Journal of Logic Programming*, 12:121–146, 1992.

[Kowalski and Sergot, 1986] Robert Kowalski and Marek Sergot, "A logic-based calculus of events," *New Generation Computing*, 4:67–95, 1986.

[Ladkin and Maddux, 1988] Peter Ladkin and R. Maddux, "Representation and reasoning with convex time intervals," Technical report KES.U.88.2, Kestrel Institution, Palo Alto, CA, 1988.

[Lifschitz, 1987] Vladimir Lifschitz, "Formal theories of action," in Frank M. Brown, editor, *Proceedings of the 1987 Workshop: The Frame Problem in Artificial Intelligence*, pages 35–58. Lawrence, KA, Morgan Kaufmann, 12–15 April 1987.

[Lifschitz and Rabinov, 1989] Vladimir Lifschitz and Arkady Rabinov, "Miracles in formal theories of action (research note)," *Artificial Intelligence*, 38(2):225–238, March 1989.

[Lin and Shoham, 1992] Fangzhen Lin and Yoav Shoham, "Concurrent actions in the situation calculus," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 580–585, San Jose, CA, 12–16 July 1992.

[Martin, 1993] Nathaniel G. Martin, *Using Statistical Inference to Plan Under Uncertainty*, PhD thesis, Department of Computer Science, University of Rochester, Rochester, NY, 1993. To appear as a Technical Report.

[Martin and Allen, 1993] Nathaniel G. Martin and James F. Allen, "Statistical Probabilities for Planning," Technical Report 474, Department of Computer Science, University of Rochester, Rochester, NY, November 1993.

[McAllester and Rosenblitt, 1991] David McAllester and David Rosenblitt, "Systematic nonlinear planning," in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 634–639, 12–19 July 1991.

[McCarthy, 1980] J. McCarthy, "Circumscription – A form of non-monotonic reasoning," *Artificial Intelligence*, 13(1,2):27–39, 1980.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. American Elsevier Publishing Co., Inc., 1969. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 393–435.

[McDermott, 1985] Drew McDermott, "Reasoning about plans," in J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, pages 269–318. Ablex Publishing, Norwood, NJ, 1985.

[Miller and Shanahan, 1994] Rob Miller and Murray Shanahan, "Narratives in the situation calculus," *Journal of Logic and Computation, Special Issue on Actions and Processes*, 1994. To appear.

[Morgenstern and Stein, 1988] Leora Morgenstern and Lynn A. Stein, "Why things go wrong: A formal theory of causal reasoning," in *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, St. Paul, MN, 21–26 August 1988. University of Minnesota. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 641–646.

[Mourelatos, 1978] A.P.D. Mourelatos, "Events, processes and states," *Linguistics and Philosophy*, 2:415–434, 1978.

[Nilsson, 1980] Nils J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1980.

[Pednault, 1986] Edwin P.D. Pednault, "Formulating multi-agent, dynamic-world problems in the classical planning framework," in M.P. Georgeff and A.L. Lansky, editors, *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Los Altos, CA, 30 June–2 July 1986. Morgan Kaufmann. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 675–710.

[Pelavin, 1991] Richard N. Pelavin, "Planning With Simultaneous Actions and External Events," in *Reasoning about Plans*, pages 127–212. Morgan Kaufmann, San Mateo, CA, 1991.

[Pinto, 1994] Javier Pinto, *Temporal Reasoning in the Situation Calculus*, PhD thesis, University of Toronto, Toronto, Ontario, Canada, February 1994.

[Reiter, 1992] Raymond Reiter, "The projection problem in the situation calculus: A soundness and completeness result, with an application to database updates," in *Proceedings of the First International Conference on AI Planning Systems*, pages 198–203, College Park, MD, 15–17 June 1992. Morgan Kaufmann.

[Sacerdoti, 1975] Earl D. Sacerdoti, "The nonlinear nature of plans," in *Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)*, pages 206–214, Tbilisi, Georgia, USSR, 3–8 September 1975. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 162–170.

[Sandewall, 1994] Erik Sandewall, *Features and Fluents*, Oxford University Press, 1994. To appear.

[Schubert, 1990] Lenhart Schubert, "Monotonic Solution of The Frame Problem in The Situation Calculus: An Efficient Method for Worlds with Fully Specified Actions," in Henry E. Kyburg, Jr., Ronald P. Loui, and Greg N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–68. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990. Also available as University of Rochester TR 306, August 1989.

[Schubert, 1992] Lenhart K. Schubert, "Explanation Closure, Action Closure, and the Sandewall Test Suite for Reasoning about Change," Technical Report 440, Department of Computer Science, University of Rochester, Rochester, NY, October 1992.

[Schubert and Hwang, 1989] Lenhart K. Schubert and Chung-Hee Hwang, "An episodic knowledge representation for narrative text," in R.J. Brachman, H.J. Levesque, and R. Reiter, editors, *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR-89)*, pages 444–458, Toronto, Ont., 15–18 May 1989.

[Shanahan, 1989] Murray Shanahan, "Prediction is deduction but explanation is abduction," in Natesa Sastri Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1055–1060, Detroit, MI, 20–25 August 1989.

[Shanahan, 1990] Murray Shanahan, "Representing continuous change in the situation calculus," in *Proceedings of the European Conference on Artificial Intelligence (ECAI-90)*, 1990.

[Shoham, 1987] Yoav Shoham, "Temporal Logics in AI: Semantical and ontological considerations," *Artificial Intelligence*, 33(1):89–104, 1987.

[Shoham, 1988] Yoav Shoham, *Reasoning about change: Time and Causation from the Standpoint of Artificial Intelligence*, MIT Press, Cambridge, MA, 1988.

[Tate, 1977] Austin Tate, "Generating project networks," in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 888–889, Cambridge, MA, 1977. MIT. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 291–296.

[Traum *et al.*, 1994] David R. Traum, James F. Allen, George Ferguson, Peter A. Heeman, Chung-Hee Hwang, Tsuneaki Kato, Nathaniel Martin, Massimo Poesio, and Lenhart K. Schubert, "Integrating Natural Language Understanding and Plan Reasoning in the TRAINS-93 Conversation System," in *Working Notes of the AAAI Spring Symposium on Active NLP*, Stanford, CA, 21–23 March 1994.

[van Benthem, 1983] Johan F. A. K. van Benthem, *The Logic of Time*, D. Reidel and Kluwer, Dordrecht and Boston, 1983.

[Vendler, 1967] Zeno Vendler, *Linguistics in Philosophy*, Cornell University Press, New York, 1967.

[Vere, 1983] Stephen A. Vere, "Planning in time: Windows and durations for activities and goals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):246–267, 1983. Also in *Readings in Planning*, J. Allen, J. Hendler, and A. Tate (eds.), Morgan Kaufmann, 1990, pp. 297–318.

[Vilain and Kautz, 1986] Marc Vilain and Henry Kautz, "Constraint Propagation Algorithms for Temporal Reasoning," in *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 377–382, Philadelphia, PA, 11–15 August 1986. University of Pennsylvania.

[Vilain *et al.*, 1990] Marc Vilain, Henry Kautz, and Peter van Beek, "Constraint propagation algorithms for temporal reasoning: A revised report," in *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufman, San Mateo, CA, 1990.

[Wilkins, 1988] David E. Wilkins, *Practical Planning: Extending the Classical AI Planning Paradigm*, Morgan Kaufmann, San Mateo, CA, 1988.