A Model of Events and Processes*

Periklis Belegrinos
Department of Computer Science
The University of Melbourne
Parkville, Victoria

Michael Georgeff
Australian Artificial Intelligence Institute
1 Grattan Street
Carlton, Victoria

Abstract

The aim of this paper is to provide a hasis for a theory of events and processes that can be used for reasoning about arbitrarily complex dynamic domains involving multiple agents. The approach is based on a model of events that explicitly represents the domain of influence of each event. By scoping an event's domain of influence, most of the problems that have plagued the more conventional stated-transition models of events can be avoided. The effect of performing events, either in isolation or concurrently with other events, is described. To represent constraints among events, a model of processes is developed. This allows the modelling of arbitrarily complex behaviours. Finally, a representation of causal influence is provided that allows the ramifications of any given event occurrence to be modelled.

1 Introduction

Representing and reasoning about events (or actions) has been a continuing problem for researchers in Artificial Intelligence (AI). What apparently makes it so hard is the difficulty in describing, in a tractable and natural way, the effects that events have — or, more particularly, do not have — on the world.

The standard approach within AI is to assume that, at any given instant, the world is in a particular *world state.* Various properties or propositions may hold of these states, A given world state has no duration; the only way the passage of time can be observed is through some change of state. The world changes its state by the occurrence of *events.*

A representation of events has to allow us to determine the outcome of performing an event in some given world state — that is, what propositions hold in the resulting world state given that certain ones hold in the initial state. The normal way to do this is to introduce a formalism, such as the situation calculus [McCarthy and Hayes, 1969), to describe world states and the propositions that hold in those states. Then one writes down certain axioms about how the performance of a given event affects the propositions holding in the world.

There are two problems here. The first is that, in any sizable domain, there are a large number of properties that do not change from one state to the next, and writing down the axioms to express these facts is simply untenable. This is what is conventionally known as the *frame problem.* The second problem is that, in most real-world domains, the effects of an event can be quite complex, depending on the situation in which the event is performed and the causal properties of the domain. This is known as the *ramification problem.* These problems become even more severe when we allow events to be performed in parallel (concurrently), as one then has to be able to specify the results of performing any event in conjunction with every other set of possible events.

Most of the recent approaches to these problems utilize nonmonotonic reasoning mechanisms. There are numerous ways this can be done, including nonmonotonic logic, circumscription, closest possible worlds (or models), and circumscriptive ignoiance. These are complex devices, and every now and then someone finds a significant flaw in one or other approach [Hanks and McDermott, 1987].

We believe that much of the problem rests with viewing events as *transitions* between world states. Clearly, the performance of an event results in the world undergoing a transition from one state to another. But, in reality, an event affects only a small part of the world, and the model of events we choose should reflect this localization property. This paper proposes a model of events and processes based on this idea.

2 Informal Description

We consider the world to consist of a set of properties or relations among individuals. A world state is defined to be an assignment of truth or falsity (or, more generally, some value) to each of these properties. We view an event or action as a black box that has certain input ports and certain output ports. The input and output ports are associated with properties of particular individuals, and the values obtaining at these ports represent the values of the associated properties. The aim is to specify the transition table for the input and output ports of the event, rather than the transition table for

"This work was supported in part by a *Generic Industry Research and Development Grant* from the Australian Department of Industry, Technology and Commerce and by Carlton and United Breweries Ltd.

the world as a whole.

To determine what state transitions are possible given an occurrence of an event, we simply require that the properties associated with the input and output ports of the event have the specified values in the initial and final states, respectively. What happens to other properties depends on what other events are, or are not, occurring at the same time.

Of course, all that we are doing here is specifying explicitly which properties are affected by an event and which are not. But this is very important, Most representations or formalisms for describing events do not explicitly specify the domain of influence of an event; instead, they determine the scope of influence by resorting to nonmonotonic mechanisms.

Such explicit specification, however, would seem to prevent solution of the ramification problem — the manner in which the occurrence of a given event affects the world can be very complex and extensive. To handle this problem, we introduce notions of causality and process. Intuitively, although the domain of influence of a particular event may be quite limited, the event can cause the occurrence of other events or processes and in this way have a far more extensive effect upon the world.

3 Modelling Events

We base our model of events on situation semantics [Barwise and Perry, 1983]. We consider a set of *individuals* or *objects I* and a set of n-ary *properties, F,* over these individuals. A *situation type* (or, simply, *situation)* is an assignment of *values* V to a set of properties from F. Each assignment in a given situation *s* is called a *constituent* of *s.* For example,

s = *{open(doorl):* t, *distance(rooml,room2):* 8}

is a situation in which *door1* is open[1] and the distance between *room1* and *room2* is 8; nothing is said about other individuals or the values of other properties.

A *state* is a complete situation where all property values for all individuals are specified. We let s# represent the null situation (i.e., the situation with no constituent). We will denote the set of all possible states in a given domain by *W.* An example of a state in the living-room world introduced by Ginsberg and Smith [1988] is given as A1 in the Appendix.

A situation *s* is *coherent* if *s* does not assign two different values to any property. Situations *s* and *t* are *compatible* if their union is coherent. A sequence of situations is called a *history.*

An *atomic event type* (or simply *event)* is a mapping from an *event name* to a pair of situation types.[2] For an event e, the first element of this pair is called the *precondition* of e, denoted *pre(e),* and the second element the *postcondition* of e, *post(e).* The properties comprising the precondition and postcondition of e are called, respectively, the domain of e, denoted *dom(e),* and the range of e, *ran(e).*

[1]The symbols t and f represent true and false, respectively,

[2] This definition can be generalized to allow sets of transitions between situation types without affecting the results given in this paper.

For example,

$o(door1) = (\{open(door1)\text{: } \mathbf{f}\}\ ,\ \{open(door1)\text{: } \mathbf{t}\})$

denotes an atomic event type representing the opening of *door1*. An example of an event from the living-room world is given as A2 in the Appendix.

We let $e_{\|} = (s_{\emptyset}, s_{\emptyset})$. If either the precondition or postcondition of an event is not coherent, we denote the event by e_{+}. Intuitively, $e_{\|}$ corresponds to a no-op and e_{+} is an event that can never be performed.

An event e can *occur* in a history between adjacent situations s and t if and only if $pre(e)$ is contained in s and $post(e)$ is contained in t. Intuitively, the precondition of the event must be satisfied in the initial situation and the postcondition must be satisfied in the resulting (final) situation.

It is important to note that the precondition of an event acts like a guard; that is, the event simply cannot occur unless the precondition is satisfied. This is different to many event formalisms in AI, in which the event can occur without the precondition being satisfied, but possibly with unpredictable results.

4 Concurrent Events

Consider two events occurring concurrently (i.e., occurring between the same two situations) in any given history. It follows directly from the condition on event occurrence given above that the preconditions of both events must be compatible and that their postconditions must be compatible. Intuitively, two events can be performed concurrently provided they do not interfere with one another.

We can also define new events consisting of the composition of two or more concurrent atomic events. More formally, we represent the concurrent combination of two atomic events e_1 and e_2, denoted by $e_1 \| e_2$, as

$$e_1 \| e_2 = (pre(e_1) \cup pre(e_2), post(e_1) \cup post(e_2)).$$

The resulting event can only occur if both its precondition and postcondition are coherent; otherwise, it reduces to the event e_{+}. An example of a concurrent event is given as A3 in the Appendix.

As the union of two situations is itself a situation, it follows that the concurrent event $e_1 \| e_2$ is also an atomic event. It can therefore be combined concurrently with other atomic events. The operator $\|$ can be shown [Belegrinos, 1991] to be reflexive, commutative and associative, with $e_{\|}$ and e_{+} playing the role of the unit and zero elements, respectively. With the above definition, we can therefore replace any set of concurrently occurring events $e_1, e_2, \ldots e_n$ by the single atomic event $e_1 \| e_2 \| \ldots \| e_n$. We denote the closure of a set of atomic events E under the operator $\|$ by $\overline{E}$.

Elsewhere [Belegrinos, 1991] we develop a method for representing more general classes of events using a greater range of event operators. For the present discussion, however, atomic events are sufficient.

5 Performing Events

We have already stated conditions that have to be satisfied for an event to occur in a particular history. We

now introduce further constraints upon events and the situations over which they occur.

Intuitively, we wish to only allow changes in properties from one situation to the next if an event occurs that brings about that change [Georgeff, 1987]. In our event representation, it is an immediate consequence that the value of any property outside the range of the events occurring between the two situations will remain unchanged.

Consider a pair of adjacent situations *s* and *t* in a given history. Let the set of all (concurrent) atomic events occurring between these situations be represented by the single atomic event e, as defined above. Then the following conditions must be observed between the situations *s* and *t:*

$$pre(e) \subset s$$
$$post(e) \subset t, \text{ and}$$
$$t|_{R-ran(e)} = s|_{R-ran(e)} \quad (1)$$

where U|Q represents the restriction of u to the properties in *Q* and *R* is the set of properties in the situation *t.*

Note that the resulting situation *t* is fully determined by the event e and the initial situation s, provided *s* includes all the properties in *t* not included in the range of e. In the particular case when *s* and t are states (i.e., complete situations), e fully specifies the transition between the states. We will denote this state by *result(e,s).* Formally, we have

$$result(e,s)|_{ran(e)} = post(e), \text{ and}$$
$$result(e,s)|_{F-ran(e)} = s|_{F-ran(e)}$$

Note that *result* is a partial function and is undefined for some states. In particular, we write *result(e,s)* = ⊥ if *pre(e)* is not compatible with *s.*

Because the domain of influence of an event is fully prescribed, describing events and their effects upon the world is not subject to the frame problem. That is, the effects of an event can be fully delineated without having to use either frame axioms or nonmonotonic mechanisms [Georgeff, 1987].

Now consider the occurrence of an event e, possibly concurrently with other events from a set of events *E.* The set of all possible transitions between states, called the *transition relation* of *e* with respect to *W* and *E,* is given by

$$T_e = \bigcup_{s \in W} T_{e,s} \text{ where } T_{e,s} = \bigcup_{d \in \overline{E}} result(e||d, s)$$

The set *Te* corresponds to the conventional view of events in the AI literature, i.e., as transition relations over world states [McDermott, 1982]. It is straightforward to prove that $T_{e_1||e_2} = T_{e_1} \cap T_{e_2}$, as expected.

While this indicates that, for every event e in a specified domain of world states *W* and events E, there is a corresponding transition relation, the converse is not true. This is quite important. It means that our notion of an event is more restrictive than notions in which arbitrary transitions are allowed. But we can represent any transition relation as a *set* of atomic events, and thus can represent any possible world history.

Moreover, the fact that our notion of events is more restrictive allows for simpler logical representations and possibly more effective reasoning techniques. For example, the model of events as proposed can be used to provide a formal semantics for the standard STRIPS representation [Fikes and Nileson, 197I] of events and is suggestive of its straightforward extension for reasoning about concurrent events. Provided that one has some means for specifying any domain-specific constraints on the occurrence of events, and extends the STRIPS representation to allow event sets and concurrency, the approach is fully general. The model proposed here can also be used for providing the formal semantics for other more powerful logics of events and actions.

6 Constraints Among Events

The above model of events is quite general and can be used to represent arbitrary histories of events and situations. However, to reason about events and their possible ramifications, we need to add to this representation some means for specifying domain-specific constraints among events.

6,1 Processes

The world can be viewed as consisting of various mechanisms or causal relationships which place constraints upon the events that can be performed. These constraints determine both which events can or must occur concurrently and which events can or must occur sequentially. Most AI formalisms have limited expressive capability for describing these kinds of causal and temporal relations among events (a notable exception being the work of Lansky [1987]). To model these constraints on behaviour in a *fully* general way, we therefore introduce the notion of *process.*

While there are a number of different process models that would be suitable to this end, we choose herein to use the model of Communicating Sequential Processes (CSP) as introduced by Hoare [1985]. Following this approach, we represent a process as a pair of sets. The first set, called the *alphabet* of the process, is the set of atomic events in which the process can engage. The alphabet of a process *P* is denoted by a*(P).* The second set, called the *traces* of the process, is a set of sequences of atomic events.

Each trace describes the behaviour of the process up to a particular point in time, and each event that appears in a trace is a possibly concurrent combination of the atomic events contained in the alphabet of the process. The traces of a process *P* are denoted by *traces(P)* and the trace that consists of the sequence of events m_1 followed by m_2 and so on up to m_n is denoted by $< m_1, m_2, \ldots, m_n >$. The first event of a trace $m \neq <>$ is denoted by m_1 and the trace which results from removing the first event of m is denoted by *m'*

This notion of process is important in a number of ways that parallel our representation of events. First, it depends on an explicit representation of the events that can possibly engage in the process, as defined by the process alphabet. Second, it is local. That is, it does not attempt to define the constraints between events as they occur globally, but restricts attention to their local influence over one another. The fact that an event e_2 directly follows another event *e* in a process *P* does not prevent

other events, not in the alphabet of *P,* from occurring between the occurrences of e_1 and e_2. These ideas are important for the definition of process concurrency, which we introduce below, and are essential to providing compositionality of the representation. Moreover, the ability to represent localized constraints on events is critical for achieving computational tractability [Lansky, 1988].

Rather than specifying the traces of a process directly, Hoare introduces an equivalent method for representing processes. This method consists of the following recursive rules.

1. The process $STOP_B$ (where B is a set of atomic events and $\alpha(STOP_B) = B$) denotes a process that never performs any of the events in B.

2. The process $e \rightarrow P$ (where $e \in \overline{\alpha(P)}$) denotes a process that performs the event e and then behaves like P.

3. The process $e_1 \rightarrow P_1 \mid e_2 \rightarrow P_2$ (where $\alpha(P_1) = \alpha(P_2)$ and $e_1 \in \overline{\alpha(P_1)}$, $e_2 \in \overline{\alpha(P_2)}$ and $e_1 \neq e_2$) denotes a process that first performs either e_1 or e_2 and then behaves like P_1 if e_1 was performed first and like P_2 if e_2 was performed first.

The above rules can be conveniently expressed using the notation $P = e : B \rightarrow P(e)$, where B is a set of atomic events and $P(e)$ is an expression denoting a process for each of the different events e in B. The process $e : B \rightarrow P(e)$ denotes a process that first performs an event $e \in B$ and then behaves like $P(e)$. The traces of this process are expressed as follows:

$$traces(e : B \rightarrow P(e)) = \{m : m = <> \text{ or } (m_1 \in B \text{ and } m' \in traces(P(m_1)))\}$$

6.2 Concurrent Processes

Different agents (or machines) and the environment in which they are situated can be modelled by different processes. It remains to specify how these processes evolve when they are brought together. For this we need the notion of concurrent processes.

The definition we adopt is similar to that of Hoare. That is, events that are unique to one process can be performed whenever that process is ready to perform them and events that are in the alphabets of more than one process must be performed simultaneously (synchronously). However, we extend Hoare's approach to allow the simultaneous occurrence of events of different types.

Formally, we represent the concurrent combination of two processes P_1 and P_2, denoted $P_1 || P_2$, as follows:

$$\alpha(P_1 || P_2) = \alpha(P_1) \cup \alpha(P_2)$$
$$traces(P_1 || P_2) = \{m : m!\alpha(P_1) \in traces(P_1) \text{ and } m!\alpha(P_2) \in traces(P_2) \text{ and } m_i \in \overline{\alpha(P_1 || P_2)} \text{ for all } m_i \text{ in } m\}$$

The expression $m!\alpha$ is the restriction of the events in m to the events in α. If no events in m are contained in α, $m!\alpha = <>$.

6.3 Relationship to Situation Histories

The notion of process as defined above allows us to represent arbitrary constraints among events[3] and to compose these in various ways. We now have to determine how these sequences of events relate to situation and state histories.

We do this by extending the notion of event occurrence between two situations within a given history. In particular, we say that an event trace can occur over a given interval within a given history just in case each element of the trace can occur between successive situations in the interval. More formally, we have e = <*e1,e2,.....,en*> can *occur* over an interval s — $S_0, S_1, \ldots, s_n$ of a history if and only if e_i can occur between situations S_{i-1} and S_i for $i = 1, \ldots, n$.[4]

If we have a set of concurrent processes *P* representing the constraints among all the events of a given domain, the traces of P will fully determine which events can occur from one moment to the next. Thus, for any such trace to occur in a particular history, the conditions given as Equation 1 in Section 5 must also be satisfied. In the case that the history is over complete states, it follows that each trace of *P* determines a unique history, given the state of the history in which *P* is initiated.

Given a set of concurrent processes *P,* we will say a trace c = < $e_1, \ldots, e_n$ > of *P* is *realizable* with respect to a state $\in W$ if and only if

$$result(e_n, (result(e_{n-1}, \ldots (result(e_1, s)) \ldots))) \neq \perp$$

where we let $result(e, \perp) = \perp$. We take the empty trace to be realizable with respect to any state.

7 Causality

One of the most important relationships between events is that of *causation.* This is particularly so in our case, because it is by this mechanism that we can represent the ramifications of any given event. In this section, we show how to model causation using processes.

Following earlier work [Georgeff, 1987], we consider two types of causation: *sequential* and *simultaneous.* Let us first consider simultaneous causation. An atomic event e_1 is said to simultaneously cause an atomic event e_2 under conditions ϕ if, whenever *e*\ occurs and the condition ϕ holds, event e_2 also occurs. This form of causation can be modelled by the following process:

$$P_C = (e_1 || e_2) \rightarrow P_C \mid (e_1 || e_{\neg\phi}) \rightarrow P_C \mid e_2 \rightarrow P_C$$

where $e_{\neg\phi}$ is an atomic event that tests if ϕ is false (i.e., has $\neg\phi$ as its precondition).

This process is such that, whenever e_1 is performed under condition ϕ, the event e_2 is forced to occur concurrently. However, if ϕ is not true, *t*\ can occur by

[3]This is not quite true. The process model given here cannot represent nondeterministic machines. However, the extension to this case, which requires the notion of *failure sets,* is relatively straightforward.

[4]A set of event traces can be considered a complex (i.e., non-atomic) event *(see* also the work of Lansky [1987]). The above definition thus defines under what circumstances such an event can occur in a history.

itself. The third alternative branch in the above process (i.e., as represented by the choice e_2) is necessary so that the occurrence of e_2 can occur unconstrained by the occurrence or otherwise of e_1.

Sequential causation models the case in which the occurrence of an event e_1 causes the subsequent occurrence of an event e_2, though possibly other events may occur between the occurrences of these events. The causation may also be dependent upon some condition o holding when e_1 is performed. This form of causation can be represented by a process similar to that given above.

Other examples of causal processes are given as A4 in the Appendix.

8 Mapping Properties to Processes

We have introduced above a model for representing events and situations and for representing arbitrarily complex relations among those events. We have defined under what conditions an event or set of events can occur (or be performed), and how such occurrences affect the situations over which the event(s) take place.

This model can be used as a basis for the semantics of various logics and calculi for reasoning about events, situations, and plans. However, in some circumstances, it is desirable to be able to reason entirely within a process framework. Therefore, in this section, we describe how to transform the current representation into an equivalent process representation.

Hoare's representation of processes is purely event based and contains no explicit representation of state properties. Determining the behaviour of a set of processes is thus simply a matter of determining how the event traces of the participating process can be combined, given the restrictions on event execution denned above. However, once we allow state properties to be introduced, events can indirectly influence one another through the effect one event has on the properties that affect performance of another. In essence, the state properties provide yet other constraints on the performance of events.

One way to represent the influence of state properties is to introduce further processes that force these constraints to be observed. We call such processes *property processes,* and introduce one for each property *p* in *F.* The property process for a property p can be viewed as an n state machine, where n is the number of values that can be assigned to *p.* Each state represents a different value of p; the current state represents the current value of p. A change in state of the machine results by performing an event e such that pre(e) is compatible with the current assignment to *p.* The resulting state of the machine is the state which represents the value specified by *post(e).*

To keep the description of the property processes relatively simple, we introduce the events d^p_v — *({p: v) , s$)* and rg = (s , {p: *v})* for each different value *v* that can be assigned to *p.* Intuitively, *d%* is an event that tests whether p has value *v* and r^p assigns the value *v* to *p.* We let the set of all such events for the property p be denoted by E_p.

Any event can be described as a concurrent combination of d^p_v and r^p_v events with different *p* and *v.* This relationship can be captured by constructing a causa] process for each event *e,* such that *e* causes the simultaneous occurrence of the events d^p (or r^p), for each valuation of *p* in the domain (range) of e.

The property process P^p for property p is then constructed as follows:

$$P^p = init^p_{v_1} \rightarrow P^p_{v_1} \mid \cdots \mid init^p_{v_n} \rightarrow P^p_{v_n}, \text{ where}$$
$$v_1, \ldots, v_n \text{ are the different assignments to } p,$$
$$init^p_{v_1}, \ldots, init^p_{v_n} \text{ are distinguished atomic events,}$$
$$\alpha(P^p) = \alpha(P^p_{v_i}) = \{init^p_{v_1}, \ldots init^p_{v_n}\} \cup E_p,$$
and where
$$P^p_{v_i} = e : C^p_{v_i} \rightarrow P^p_{v_i}(e),$$
$$C^p_{v_i} = C^p_{v_{i1}} \cup \cdots \cup C^p_{v_{in}}$$
$$C^p_{v_{ii}} = \{d^p_{v_i}, r^p_{v_i}, d^p_{v_i} \| r^p_{v_i}\}$$
$$C^p_{v_{ij}} = \{r^p_{v_j}, d^p_{v_i} \| r^p_{v_j}\} \ (j \neq i)$$
and
$$P^p_{v_i}(e) = P^p_{v_j} \text{ if } e \in C^p_{v_{ij}}$$

The purpose of the events $init^p_{v_i}$ is simply to set the initial state of the property process; their form is straightforward and is given elsewhere [Belegrinos, 1991]. Intuitively, $P^p_{v_i}$ represents the state in which *p* has the value v_i and the events in $C^p_{v_{ij}}$ represent those events $e \in \overline{E_p}$ that change the value of the property *p* from v_i to v_j. Note that these processes correspond directly to the constraints used by Lansky [1987] to represent event properties, with the events r^p_v playing the role of Lansky's "adders" (v = t) and "deleters" (v = f) for property *p* and d^p_v representing the constraints on event preconditions.

Given an arbitrary process *P* and a set of property processes as described above, it can be shown [Belegrinos, 1991] that the traces of *P* executed concurrently with the property processes are exactly those traces that would result from executing *P*, taking into account the influence of state properties. That is, the traces are exactly the realizable traces of *P*.

9 Conclusions

The aim of this paper has been to determine the basic foundations for a theory of events, situations, and processes that can be used for reasoning about arbitrarily complex dynamic domains. The essential elements of the approach are: (I) A model of events that explicitly represents the domain of influence of the event; (2) A model of processes that can represent arbitrarily complex behaviours; (3) An extension of that process model to represent the influence of events upon state properties and the influence, in turn, of those properties upon other events; and (4) A representation of causality that allows the ramifications of a given event occurrence to be modelled.

We have also shown how to map the representation of a problem domain described in terms of high-level events, state properties, and causality into a more-or-less standard process model. Having done this, we can then draw on all the work in process theory and its application to help solve complex, real-world problems.

Although we have not directly addressed the frame problem, it should be clear that, by scoping the domain of influence of events, the problem as conventionally viewed will not arise. In essence, we have transferred the problem of reasoning about the effects of events and actions to one of reasoning about which events are occurring at any given moment of time. While we believe that process models provide the basis for doing this reasoning in a tractable way, this claim has yet to be demonstrated. We also need to develop a logic of events and situations that is well suited to this model and to examine the semantic foundations, in terms of this model, of previous formalisms.

Appendix

We illustrate here how our approach can be used to reason about the living-room world introduced by Ginsberg and Smith [1988]. For simplicity, we consider only a subset of this world. In particular, we assume that the room contains a television, a plant, a shelf, and two ventilation ducts located under the floor. If an object is on a duct, that duct is blocked; if both ducts are blocked, the room will become stuffy. We can represent this world by taking

$$F = \{pos(\text{tv}),\ pos(\text{plant}),\ blocked(\text{duct1}),\ blocked(\text{duct2}),\ stuffy(\text{room})\},$$

where
pos can take the values $\{floor, shelf, duct1, duct2\}$ and *blocked* and *stuffy* the values $\{\text{t}, \text{f}\}$.

The initial state of the world is represented by the state s as follows:

$$s = \{pos(\text{tv}){:}\ \text{shelf}\ ,\ pos(\text{plant}){:}\ \text{duct2}\ ,\ blocked(\text{duct1}){:}\ \text{f}\ ,\ blocked(\text{duct2}){:}\ \text{t}\ ,\ stuffy(\text{room}){:}\ \text{f}\} \quad \text{(A1)}$$

Suppose we can perform two events in this world: move the television from the shelf to duct1 (denoted by e_1) and move the plant from duct2 to the floor (denoted by e_2). We can represent these events and their concurrent combination as follows:

$$e_1 = (\{pos(\text{tv}){:}\ \text{shelf}\ ,\ blocked(\text{duct1}){:}\ \text{f}\}\ ,\ \{pos(\text{tv}){:}\ \text{duct1}\ ,\ blocked(\text{duct1}){:}\ \text{t}\})$$
$$e_2 = (\{pos(\text{plant}){:}\ \text{duct2}\ ,\ blocked(\text{duct2}){:}\ \text{t}\}\ ,\ \{pos(\text{plant}){:}\ \text{floor}\ ,\ blocked(\text{duct2}){:}\ \text{f}\}) \quad \text{(A2)}$$

$$e_1 \| e_2 = (\{pos(\text{tv}){:}\ \text{shelf}\ ,\ blocked(\text{duct1}){:}\ \text{f}\ ,\ pos(\text{plant}){:}\ \text{duct2}\ ,\ blocked(\text{duct2}){:}\ \text{t}\}\ ,\ \{pos(\text{tv}){:}\ \text{duct1}\ ,\ blocked(\text{duct1}){:}\ \text{t}\ ,\ pos(\text{plant}){:}\ \text{floor}\ ,\ blocked(\text{duct2}){:}\ \text{f}\}) \quad \text{(A3)}$$

Let P_A be an agent process that performs the event e_1 followed by e_2. That is, $\alpha(P_A) = \{e_1, e_2\}$ and $P_A = e_1 \rightarrow e_2 \rightarrow STOP_{\alpha(P_A)}$.

The indirect effects (ramifications) of these events on the property *stuffy*(room) are handled by introducing the events e_{stuffy} and e_{clear}, defined as follows:

$$e_{clear} = (s_\emptyset\ ,\ \{stuffy(\text{room}){:}\ \text{f}\})$$
$$e_{stuffy} = (s_\emptyset\ ,\ \{stuffy(\text{room}){:}\ \text{t}\})$$

Depending upon which causal rules one adopts, we can force the room to become stuffy immediately the two ducts are blocked or some time after both are blocked. For this example, we will assume the former. This leads to the following causal processes:

$$P_{C_1} = (e_2 \| e_{clear}) \rightarrow P_{C_1}$$
$$P_{C_2} = (e_1 \| e_\phi \| e_{stuffy}) \rightarrow P_{C_2}\ |\ (e_1 \| e_{\neg\phi}) \rightarrow P_{C_2}$$

where

$$e_\phi = (\{blocked(\text{duct2}){:}\ \text{t}\}\ ,\ s_\emptyset)$$
$$e_{\neg\phi} = (\{blocked(\text{duct2}){:}\ \text{f}\}\ ,\ s_\emptyset) \quad \text{(A4)}$$

Putting these processes together, it is not too hard to show that $< init_s, (e_1 \| e_\phi \| e_{stuffy}), (e_2 \| e_{clear}) >$ is the only realizable trace of P, where $init_s$ is the event that sets the initial state properties to s.

Acknowledgments

The authors wish to thank David Morley, Anand Rao, and Liz Sonenberg for their contributions to this work.

References

[Barwise and Perry, 1983] J. Barwise and J. Perry. *Situations and Attitudes.* MIT Press, Cambridge, Massachusetts, 1983.

[Belegrinos, 199l] P. Belegrinos. A Model of Actions and Processes. Technical Report 91/6, The University of Melbourne, Parkville, Victoria, 1991.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence,* 2:189-208, 1971.

[Georgeff, 1987] M, P. GeorgerT. Actions, Processes, and Causality. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop,* pages 99-122. Morgan Kaufmann, Mountain View, California, 1987.

[Ginsberg and Smith, 1988] M. L, Ginsberg and D, E, Smith, Reasoning about Action I: A Possible Worlds Approach, *Artificial Intelligence,* 35:165-195, 1988.

[Hanks and McDermott, 1987] S. Hanks and D. McDermott. Nonmonotonic Logic and Temporal Projection, *Artificial Intelligence,* 33:379-412, 1987.

[Hoare, 1985] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, UK, 1985.

(Lansky, 1987] A. L, Lansky. A Representation of Parallel Activity Based on Events, Structure, and Causality. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop,* pages 123-159. Morgan Kaufmann, Mountain View, California, 1987.

[Lansky, 1988] A. L. Lansky. Localized Event-Based Reasoning for Multiagent Domains. *Computational Intelligence,* 4:319-340, 1988.

[McCarthy and Hayes, 1969] J. McCarthy and P. J, Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence,* 4:463-502, 1969.

[McDermott, 1982] D. McDermott. A Temporal Logic for Reasoning about Processes and Plans. *Cognitive Science,* 6:101-155, 1982.