A Theory of Action for MultiAgent Planning

Michael Georgeff
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025.

Abstract

A theory of action suitable for reasoning about events in multiagent or dynamically changing environments is presented. A device called a process model is used to represent the observable behavior of an agent in performing an action. This model is more general than previous models of action, allowing sequencing, selection, nondeterminism, iteration, and parallelism to be represented. It is shown how this model can be utilized in synthesizing plans and reasoning about concurrency. In particular, conditions are derived for determining whether or not concurrent actions are free from mutual interference. It is also indicated how this theory provides a basis for understanding and reasoning about action sentences in both natural and programming languages.

1. Introduction

If intelligent agents are to act rationally, they need to be able to reason about the effects of their actions. Furthermore, if the environment is dynamic, or includes other agents, they need to reason about the interaction between their actions and events in the environment, and must be able to synchronize their activities to achieve their goals.

Most previous work in action planning has assumed a single agent acting in a static world. In such cases, it is sufficient to represent actions as state change operators (e.g., [4], [9]). However, as in the study of the semantics of programming languages, the interpretation of actions as functions or relations breaks down when multiple actions can be performed concurrently. The problem is that, to reason about the effects of concurrent actions, we need to know *how* the actions are performed, not just their final effects.

Some attempts have recently been made to provide a better underlying theory for actions. McDermott [10] considers an action or event to be a set of sequences of states, and describes a temporal logic for reasoning about such actions and events. Allen [1] also considers an action to be a set of sequences of states, and specifies an action by giving the relationships among the intervals over which the action's conditions and effects are assumed to hold. However, while it is possible to state arbitrary properties of actions and events, it is not obvious how one could use these logics in synthesizing or verifying multiagent plans. [1]

In a previous paper [5], we proposed a method for forming synchronized plans that allowed multiple agents to achieve multiple goals, given a simple model of the manner in which the actions of one agent interact with those of other agents. In this paper, we propose a more general model of action, and show how it can be used in the synthesis or verification of multiagent plans and concurrent programs.

2. Process Models and Actions

Agents are machines or beings that act in a world. We distinguish between the internal workings of an agent and the external world that affects, and is affected by, that agent. All that can be observed is the external world. At any given instant, the world is in a particular ***world state***, which can be described by specifying conditions that are true of that state.

Let us assume that the world develops through time by undergoing discrete changes of state. Some of these changes are caused by agents acting in the world; others occur "naturally," perhaps as a result of previous state changes. Actions and events are considered to be composed of primitive objects called ***atomic transitions***. An atomic transition is a relation on the set of world states. Any sequence of states resulting from the application of some specified atomic transitions will be called an ***event***. Note that we do not require that atomic transitions be deterministic, but we do require that they terminate.

An *action* is a class of events; viewed intuitively, those that result from the activity of some agent or agents in accomplishing some goal (including the achievement of desired conditions, the maintenance of desired invariants, the prevention of other events, etc.)

In carrying out or performing an action, an agent forces some sequence of atomic transitions in the world. For every action the agent is capable of performing, there will correspond some internal structure that specifies just how and under what conditions these atomic transitions are to be made.

[1] Allen [2] proposes a method for forming multiagent plans that is based on his representation of actions. However, he does not use the temporal logic directly, and actions are restricted to a particularly simple form (e.g., they do not include conditionals).

Usually we do not have access to this internal structure. However, since we are interested only in the ***observable*** behavior of the agent, we do not need to know the internal processes that govern the agent's actions. Thus, to reason about how the agent acts in the world and how these actions interact with events in the world, we need only an abstract model that explains the observable behavior of the agent.

We shall specify the class of possible and observable behaviors of an agent when it performs an action by means of a device called a ***process model.*** A process model consists of a number of internal states called ***control points.*** At any moment in time, ***execution*** can be at any one of these control points. Associated with each control point is a *correctness condition* that specifies the allowable states of the world at that control point.

The manner in which the device performs an action is described by a partial function, called the ***process control function***, which, for a given control point and given atomic transition, determines the next control point. A process model can thus be viewed as a finite-state transition graph whose nodes are control points and whose arcs are labeled with atomic transitions.

A process model for an action stands in the same relationship to the internal workings of an agent and events in the external world as a grammar for a natural language bears to the internal linguistic structures of a speaker and the language that is spoken. That is, it models the observable behavior of the agent, without our claiming that the agent actually possesses or uses such a model to generate behaviors.

3. Formal Definition

A *process model* describes an action open to an agent. Formally, a process model is a seven-tuple $\mathcal{A} = \langle S, F, C, \delta, P, c_I, c_F \rangle$ where

- S is a set of ***world states***
- $F : SXS$ is a set of ***atomic transitions***
- C is a set of ***control points***
- $\delta : C \text{ X } F \rightarrow C$ is a ***process control function***
- $P : C \rightarrow 2^S$ associates subsets of S with each control point; values of this function are called ***correctness** conditions*
- $c_I \in C$ is the ***initial control point***
- $c_F \in C$ is the ***final control point.***

In general, δ is a partial function. If for a control point c and atomic transition tr, $\langle c, tr \rangle$ is in the domain of δ, we say that tr is *applicable* at c.

We are now in a position to define the execution of a process model. Let $\mathcal{A}$ be a process model as defined above. We first define a *state of execution* of $\mathcal{A}$ to be a pair $\langle u, c \rangle$, where [2] $c \in C$ and $u \in S^*$. We say that a state of execution $e_1 = \langle us_1, c_1 \rangle$ *directly generates* a state of execution $e_2 = \langle us_1 s_2, c_2 \rangle$, denoted $e_1 \rhd_{\mathcal{A}} e_2$, if either

1. $\exists tr \, . \, \delta(c_1, tr) = c_2$ and $\langle s_1, s_2 \rangle \in tr$, or
2. $c_1 = c_2$

In (1) we say that the transition is effected by the agent executing $\mathcal{A}$, while in (2) we say that the transition is effected by the *environment.*

We now define a restriction on the relation $\rhd_{\mathcal{A}}$. If, for e_1 and e_2 defined above, $e_1 \rhd_{\mathcal{A}} e_2$ and $s_2 \in P(c_2)$, we say that e_1 *successfully* generates e_2, denoted $e_1 \Rightarrow_{\mathcal{A}} e_2$. If $s_2 \notin P(c_2)$, execution is said to ***fail.***

Let $\Rightarrow_{\mathcal{A}}^*$ denote the reflexive transitive closure of the relation $\Rightarrow_{\mathcal{A}}$. Then the action generated by $\mathcal{A}$, denoted $\alpha_{\mathcal{A}}$, is defined to be

$$\alpha_{\mathcal{A}} = \{ b \mid \langle s, c_I \rangle \Rightarrow_{\mathcal{A}}^* \langle b, c_F \rangle \text{ and } s \in P(c_I) \}$$

Each element of $\alpha_{\mathcal{A}}$ is called a ***behavior*** or ***act*** of $\mathcal{A}$. The action α itself is the set of all behaviors resulting from the execution of $\mathcal{A}$.

Viewed intuitively, the device works as follows. If it is at control point c_1 and the world is in a state s_1 satisfying the correctness condition $P(c_1)$, the device can pass to control point c_2 and the world to state s_2 as long as there exists an applicable atomic transition tr between states s_1 and s_2 and $\delta(c_1, tr) = c_2$. *Alternatively,* the device can stay at control point c_1 and some transition or event occur in the world (perhaps resulting from the action of some other agent). In either case, for the execution to be successful (not to fail), the new world state must satisfy the correctness condition at c_2, i.e., s_2 must be an element of $P(c_2)$.

In performing the action α, the device starts at control point c_I. The action terminates when the device reaches c_F. Given an initial state of the world s, various sequences of world states can be generated by the process model as it passes from the initial to the final control point. The set of all such sequences constitute the action itself.

This is the same general view of action as presented by Allen [1] and McDermott [10]. However, our theory differs in that it allows us to distinguish between transitions effected by the agent and those effected by the external world. This is particularly important in the synthesis and verification of multiagent plans and concurrent programs (e.g., [3]).

Note that we do not require that a state satisfying the correctness condition at a control point be in the domain of some atomic transition applicable at that control point. Thus, it is possible for the agent to arrive at an intermediate control point and not to be able to immediately effect a further transition. In such cases, the environment must change before the action can progress. This could occur, for example, if an agent nailing two boards together expected another to help by holding the boards. Only when the "holder" (who is part of the environment) has provided the necessary assistance (and moved the state of the world into

[2] S^* is the set of all finite sequences over S.

the domain of an applicable transition) can the "nailer" proceed with the action.

Neither do we require that an atomic transition performed by an agent always be successful i.e., the transition could sometimes leave the agent in a state that violated the current correctness condition. A process model that allowed such transitions could sometimes fail. In most cases, this is undesirable (though it may be unavoidable), and for the rest of the paper we will assume that this ***cannot*** happen. That is, we will assume that only the environment (or another agent) can cause an action to fail.

It should also be noted that the correctness conditions say nothing about termination — it may be that an action never reaches completion. This can be the case if the action is waiting for a condition to be satisfied by the environment (so that a transition can be effected), it loops forever, or the environment is *unfair* (i.e., does not give the action a chance to execute).

In many cases, we wish to model actions that proceed at an undetermined rate and fail if they are ever forced to suspend execution. For example, it is difficult to hit a golf ball if the environment is allowed to remove and replace the ball at arbitrary times during one's swing. Such uninterruptable actions require that, for any control point c, any state that satisfies the correctness condition at c also be in the domain of some atomic transition applicable at c.

4. Composition of Actions

A plan or program for an agent is a syntactic object consisting of primitive operations combined by constructions that represent sequencing, nondeterministic choice, iteration, forks and joins, etc. If we intend the denotations of such plans to be process models, we need some means of combining the latter in a way that reflects the composition operators in plans.

Of special interest, and indeed the motivation behind the model presented here, is the parallel-composition operator. We define this below.

Let $\mathcal{A}_1 = \langle S, F_1, C_1, \delta_1, P_1, c_{I1}, c_{F1}\rangle$ and $\mathcal{A}_2 = \langle S, F_2, C_2, \delta_2, P_2, c_{I2}, c_{F2}\rangle$ be two process models for actions α_1 and α_2, respectively. Then we define a process model representing the parallel composition of $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted $\mathcal{A}_1 \parallel \mathcal{A}_2$, to be the process model $\langle S, F, C, \delta, P, c_I, c_F\rangle$, where

- $F = F_1 \cup F_2$
- $C = C_1 \; X \; C_2$
- For all $c_1 \in C_1$, $c_2 \in C_2$ and tr in F_1, $\delta(\langle c_1, c_2\rangle, tr) = \langle \delta_1(c_1, tr), c_2\rangle$
- For all $c_1 \in C_1$, $c_2 \in C_2$ and tr in F_2, $\delta(\langle c_1, c_2\rangle, tr) = \langle c_1, \delta_2(c_2, tr)\rangle$
- For all $c_1 \in C_1$ and $c_2 \in C_2$, $P(\langle c_1, c_2\rangle) = P_1(c_1) \cap P_2(c_2)$
- $c_I = \langle c_{I1}, c_{I2}\rangle$
- $c_F = \langle c_{F1}, c_{F2}\rangle$

It is not difficult to show that the action α generated by $\mathcal{A}_1 \parallel \mathcal{A}_2$ is exactly $\{x \cap y \mid x \in \alpha_1 \text{ and } y \in \alpha_2\}$.

Note that the projection of δ onto C_1 and C_2 gives exactly the control function for the component process models. At any moment, each component is at one of its own control points; the pair of control points, taken together, represents the current control point of the parallel process.

Furthermore, the behaviors generated by these two processes running in parallel are also generated by each of them running separately. This means that any property of the behaviors of the independent processes can be used to determine the effect of the actions running in parallel. This is particularly important in providing a compositional logic for reasoning about such actions (see [3]).

The above model of parallel execution is an interleaving model. Such a model is adequate for representing almost all concurrent systems. The reason is that, in almost all cases, it is possible to decompose actions into more and more atomic transitions until the interleaving of transitions models the system's concurrency accurately. The nondeterministic form of the interleaving means that we make no assumption about the relative speeds of the actions. We can also define a parallel composition operator that is based on communication models of parallel action, in which communication acts are allowed to take place ***simultaneously***. This, together with other composition operators, is described by me elsewhere [6].

5. Freedom from Interference

In plan synthesis and verification it is important to be able to determine whether or not concurrent actions interfere with one another. In the previous section we defined what it meant for two actions (strictly speaking, process models) to run in parallel. Now we have to determine whether execution of such a parallel process model could *fail* because of interaction between the two component processes.

Consider, then, two actions α and β generated by process models $\mathcal{A}$ and $\mathcal{B}$, respectively. The process model corresponding to these actions being performed in parallel is $\mathcal{A} \parallel \mathcal{B}$. In analysing this model, however, we will view it in terms of its two component process models (i.e., $\mathcal{A}$ and $\mathcal{B}$).

Assume that we are at control points c_1 in $\mathcal{A}$ and c_2 in $\mathcal{B}$, and that tr is an atomic transition applicable at c_2. Clearly, if the process has not failed, the current world state must satisfy both $P(c_1)$ and $P(c_2)$. Now assume that process $\mathcal{B}$ continues by executing the atomic transition tr. This transition will take us to a new world state, while leaving us at the same control point within $\mathcal{A}$. From $\mathcal{A}$'s point of view, this new state must still satisfy the condition $P(c_1)$. Thus, we can conclude that the transition tr executed at control point c_2 will not cause $\mathcal{A}$ to fail at c_1 if the following condition holds:

$$\forall s_1 \, s_2 \, . \, s_1 \in P(c_1) \cap P(c_2) \text{ and } \langle s_1, s_2\rangle \in tr \text{ implies } s_2 \in P(c_1)$$

We say that the transition tr at control point c_2 ***does not interfere with*** $\mathcal{A}$ if the above condition holds at all control points in $\mathcal{A}$, i.e., for all correctness conditions associated with $\mathcal{A}$.

We are now in a position to define freedom from interference. A set of process models $\mathcal{A}_1, \ldots \mathcal{A}_n$ is said to be *interference-free*[3] if the following holds for each process $\mathcal{A}_i$: for all control points c in $\mathcal{A}_i$ and all transitions tr applicable at c and for all $j, j \neq i$, tr at c does not interfere with $\mathcal{A}_j$.

Thus, if some set of actions is interference-free, none can be caused to fail because of interaction with the others. Of course, any of the actions could fail as a result of interaction with the environment.

From this it follows that, for ascertaining freedom from interference, it is sufficient to represent the functioning of a device by

1. A set of correctness conditions, and
2. A set of atomic transitions restricted to the correctness condition of the node from which they exit.

Knowledge of a process model's structure (i.e., the process control function), is unnecessary for this purpose. In a distributed system, this means that an agent need only make known the foregoing information to enable it to interact safely with other agents. We call such information a *reduced specification* of the action.

Let us consider the following example. Blocks A, B and C are currently on the floor. We wish to get blocks A and B on a table, and block C on a shelf, and have two agents, X and Y, for achieving this goal. Agent X has not got access to block B, but can place block A on the table and block C on the shelf. He therefore forms a plan for doing so. Agent Y cannot reach block A, but is happy to help with block B. Unfortunately, in doing so, he insists that the floor be clear of block C at the completion of his action.

The plans for agent X and Y are given below. The correctness conditions at each control point in the plans are shown in braces, "{" and "}". The "if" statement is assumed to be realized by two atomic transitions. The first of these is applicable when block C is on the floor, and results in block C being placed on the table. The second is applicable when block C is not on the floor, and does nothing (i.e., is a no-op). The process models corresponding to these plans should be self-evident.

Plan for agent X:

```
{(clear A) and (clear C)}
(puton A TABLE)
{(on A TABLE) and (clear C)}
(puton C SHELF)
{(on A TABLE) and (on C SHELF)}
```

Plan for agent Y:

```
{(clear B) and (clear C)}
(puton B TABLE)
{(on B TABLE) and (clear C)}
if (on C FLOOR) then
                (puton C TABLE)
{(on B TABLE) and not (on C FLOOR)}
```

It is clear from the definition given above that these actions are interference-free. However, they interact in quite a complex manner. In some circumstances, agent Y *will* put block C on the table, which would seem to suggest interference. Nevertheless, interference freedom is assured because the only time that Y can do this is when it does not matter, i.e., before X has attempted to put C on the shelf. Note that if the test and action parts of the "if" statement were separate atomic transitions, rather than a single one, then the actions would not be free from interference.

6. General Reasoning about Actions

So far we have been interested solely in reasoning about possible interference among actions. For many applications, we may wish to reason more generally about actions. One way to do this is to construct a logic suitable for reasoning about process models and the behaviors they generate. That is, we let process models serve as interpretations for plans or programs in the logic. An interesting compositional temporal logic has been developed by Barringer et al [3]. Because it is compositional, process models provide a natural interpretation for the logic.

One may well ask what role process models play, given that the only observables are sequences of world states and that a suitable temporal logic, per se, is adequate for describing such sequences. However, in *planning* to achieve some goal, or *synthesizing* a program, we are required to do more than just describe an action in an arbitrary way — we must somehow form an object that allows us to choose our *next* action (or atomic transition) purely on the basis of the current execution state, without any need for further reasoning.

We could do this by producing a temporal assertion about the action from which, at any moment of time, we could directly ascertain the next operation to perform (e.g., a formula consisting of appropriately nested "next" operators). Thus, in a pure temporal logic formalism, plan synthesis would require finding an approriately structured temporal formula from which it was possible to deduce satisfaction of the plan specification. However, instead of viewing planning syntactically (i.e., as finding temporal formulas with certain structural properties), it is preferable, and more intuitive, to have a model (such as a process model) that explicitly represents the denotation of a plan or program (see [6]).

Process models serve other purposes also. For example, interference freedom is easily determined, given a process model, but it is less clear how this could be achieved ef-

[3]This definition of the notion "interference-free" generalizes to arbitrary transitions that used by Owicki and Gries[11] for verifying concurrent programs. Synchronization primitives have not been included explicitly, but can be handled by conditional atomic transitions [8].

ficiently, given a general specification in a temporal logic. Even so, one would need to construct an appropriate process model first (or its syntactic equivalent in a temporal logic), as the implementation of the specifications might make it necessary to place additional constraints upon the plan.

In combination with a temporal logic such as suggested above, the proposed theory of action provides a semantic basis for commonsense reasoning and natural-language understanding. Process models are more general than previously proposed models (e.g., [7]), particularly in the way they allow parallel composition. They can represent most actions describable in English, including those that are problematic when actions are viewed as simple state-change operators, such as "walking to the store while juggling three balls" [1], "running around a track three times" [10], or "balancing a ball" (which requires a very complex process model despite the apparent simplicity of its temporal specification). The theory also allows one to make sense of such notions as "sameness" of actions, incomplete actions (like an interrupted painting of a picture) and other important issues in natural-language understanding and commonsense reasoning.

Process models are also suitable for representing most programming constructs, including sequencing, nondeterministic choice (including conditionals) and iteration. Parallelism can also be represented, using either an interleaving model, as described in section 4, or a communication model. The model used by Owicki and Gries [11] to describe the semantics of concurrent programs can be considered a special case of that proposed herein.

7. Conclusions

A nascent theory of action suitable for reasoning about interaction in multiagent or dynamically changing environments has been presented. More general than previous theories of action, this theory provides a semantics for action statements in both natural and programming languages.

The theory is based on a device called a process model, which is used to represent the observable behavior of an agent in performing an action. It was shown how this model can be utilized for reasoning about multiagent plans and concurrent programs. In particular, a parallel-composition operator was defined, and conditions for determining freedom from interference for concurrent actions were derived. The use of process models as interpretations of temporal logics suitable for reasoning about plans and programs was also indicated.

REFERENCES

[1] Allen, J. F., "A General Model of Action and Time," Comp Sci Report TR 97, University of Rochester (1981).

[2] Allen, J.F., "Maintaining Knowledge about Temporal Intervals," *Comm. ACM*, Vol 26, pp. 832-843 (1983).

[3] Barringer, H., Kuiper, R., and Pnueli, A., "Now You May Compose Temporal Logic Specifications" (1984).

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

[5] Georgeff, M.P., "Communication and Interaction in Multiagent Planning," *Proc. AAAI-83*, pp. 125-129 (1983).

[6] Georgeff, M.P., "A Theory of Plans and Actions," SRI AIC Technical Report, Menlo Park, California (1984).

[7] Hendrix, G.G., "Modeling Simultaneous Actions and Continuous Processes," *Artificial Intelligence*, Vol 4 (1973).

[8] Lamport, L., and Schneider, F.B., "The "Hoare Logic" of CSP, and All That", ACM Transactions on Programming Languages, Vol 6, pp 281-296 (1984).

[9] McCarthy, J., "Programs with Common Sense," in *Semantic Information Processing* M. Minsky ed. (MIT Press, Cambridge, Massachusetts) (1968).

[10] McDermott, D., "A Temporal Logic for Reasoning about Plans and Processes," Comp. Sci. Research Report 196, Yale University (1981).

[11] Owicki, S. and Gries, D., "Verifying Properties of Parallel Programs: An Axiomatic Approach," *Comm. ACM*, Vol 19, pp 279-285 (1976).