Actions as processes: a position on planning

Pym, David; Pryor, Louise; Murphy, David

Technical Report No. 696

Department of Computer Science

Actions as processes: a position on planning

David J. Pym
Louise M. Pryor
David V.J. Murphy

Queen Mary and Westfield College

UNIVERSITY OF LONDON

January 1995

696

To appear: Working Notes of the AAAI Spring Symposium on
Extending Theories of Action: Formal Theory and
Practical Applications, Stanford University, 1995.

Actions as processes: a position on planning

David Pym
University of London

Louise Pryor
University of Edinburgh

David Murphy
University of Birmingham*

Introduction

The world is a dynamic place. Things change because of our actions and because of others' actions. Moreover, we have no privileged status: others are able to act without our permission in unexpected ways, without explanation.

We contend that it is both natural and necessary to give an account of planning in such a world. Classical theories of planning are essentially static — they assume that the world changes only as a result of the planner's actions, that it has complete knowledge of the consequences of its actions and that the environment is predictable. Thus we contend that classical theories of planning are inadequate for domains with dynamic environments and evolving plans — essential features of problem-solving in the real world.

We propose a solution based on the *algebraic theory of processes* that shifts the focus of attention in the account of planning from *changes of state* to the *processes* by which transitions occur. One can ask what actions are required to bring about a given change of state; one can also ask what changes have been brought about by a given action. What must Dr. Who [11] do in order to conquer the Daleks? What happens if the Doctor attempts to destroy the Daleks? The description of complex actions via the algebraic theory of processes allows one to see that the latter is superior to the former.

Locally, a state-based description of a planning problem must work with a *fixed granularity of representation*. Let us suppose that two "closest" states are the two ends of a corridor, guarded by a Dalek, that Dr. Who plans to traverse. In the state-based description, the Doctor will either succeed or fail, depending on whether or not he is hit by the Dalek's ray as he makes the transition from his starting state to his finishing state. In a process-based analysis, an attempt to traverse the corridor is described by a combination of actions that, for example, tell the Doctor to run short distances at a time, hide behind pillars and keep observing the environment. This last possibility provides an indication of the key advantage of our approach: on the way down the corridor, the Doctor reasons that he's not going to make it. However, he *observes* an air-duct in the wall and realizes that he can escape down it. He takes this route and modifies his plan accordingly. We remark that it is important that basic actions be combined in such a way that they occur *concurrently* and can be *interrupted* (so that the observation of the duct can be utilized to the Doctor's advantage). Whilst it is clear that the level of state-granularity chosen in this example (the ends of the corridor) is unrealistic, we can produce a similar example for any given level of granularity. The inherently concurrent nature of the description of the plan (the Doctor runs and observes at the same time) means that action-granularity will cause no corresponding difficulties (since the Doctor is always observing, he can always be deflected).[1]

Thus we contend that actions must be represented in order that *plans*, complex

*Supported by U.K. EPSRC Research Fellowship.

[1] There is a subtle point here concerning the distinction between *interleaving* and *non-interleaving* semantics for concurrent processes.

combinations of actions into processes, can be represented: plans are represented so that an agent can execute them. Our discussion therefore begins by considering what features an action representation must have in order to account for plan execution in a dynamic environment. This discussion naturally suggests an account based on the theory of processes: we proceed by comparing this account with the more conventional approach based on state-transitions. Finally, we discuss the kinds of logics of change that are required in our setting, emphasizing the vital distinction between the reasoning performed by an agent and that performed by an agent's designer.

Agents in environments

Consider an agent executing a plan in an environment. The agent can observe and interact with the environment locally, but in general can predict neither the environment's nor its own development: many joint evolutions are possible. We suggest three reasons for this state of affairs:

(a) The environment may be too *complex* for the agent to be able to maintain or even acquire a faithful representation of it. For example, even the Doctor cannot be aware of, let alone keep track of, every one of the hundreds of Daleks whose ship he is trying to disable;

(b) The environment may be *dynamic*, evolving over time in unpredictable ways as a result not only of the agent's actions but also of those of cohabiting agents and of exogenous influences;

(c) The agent's own actions may be *nondeterministic* in that their results may not be wholly determined in a given situation.

Any theory of representation must therefore recognize the essentially non-privileged status of agents in the world; in particular, our agent's own actions should be represented in the same way as external actions. A theory must also be robust enough to cope with the operational requirements listed above. Our analysis takes these factors into account, and is also motivated by two further concerns:

(d) That the theory include an account of our agent's reasoning: we want to be able to reason about *how* the Doctor formulates *his* strategy;

(e) That both plan construction and execution be susceptible to precise *metatheoretic behavioural analyses*: for example, under what conditions will a given strategy be invoked? When will it work?

We contend that the algebraic theory of processes [7] provides a framework that meets these requirements. In process algebra, which plays a central rôle in concurrency theory, the possible behaviours of an agent and its environment are specified as *separate, interacting processes*. The joint behaviour of the agent in the environment can then be inferred from those of the individual processes and their interactions.

A process represents the totality of an agent's or environment's possible behaviour. In effect, an agent's plans are represented by the processes that describe its behaviour.[2] We define processes using a syntax of *process combinators*: these are the basic "logical constants" that are used to build up behavioural descriptions from simple components. A fragment of the syntax is[3]

$$P \quad ::= \quad \textsc{Nil} \mid a \cdot P \mid P_1 \oplus P_2 \mid P_1 + P_2.$$

Here NIL is the empty agent that does nothing, $a \cdot P$ is an agent that performs an action a and then behaves like P, and the two choice operators $+$ and $\oplus$ offer two different kinds of choice between P_1 and P_2. For example, suppose the Doctor tosses a coin to decide whether to risk himself or

[2] For now, we shall use the terms "agent" and "plan" interchangeably to denote the behaviour of an agent executing a plan.

[3] A full treatment would include combinators for parallel composition (allowing us to deal with communities of interacting agents), recursion, and others. We could also consider further features of execution of actions, including their possible duration, non-terminating actions, and so on.

not. He cannot influence whether the coin comes down heads or tails; this is decided by the environment. We write his plan as $(Heads \cdot \text{RISK}) + (Tails \cdot \text{SAFE})$, where $+$ indicates an *external* choice: from the Doctor's point of view, the choice is made by the environment; once the environment has acted, by making the coin come down heads or tails, his course of action has been decided.

A description of the environment, in contrast, models the fact that it is it, rather than the Doctor, that decides how the coin lands. Thus we write the process that models the environment as $(Heads \cdot \text{NIL}) \oplus (Tails \cdot \text{NIL})$, where $\oplus$ indicates an *internal choice*: the environment makes this choice without reference to external factors.

Processes versus state-transitions

Historically, there have been two main approaches to action representation in planning: the situation calculus [14] and STRIPS add-and-delete lists [1]. These approaches, and others that derive from them, are based on the notion of action as a transition between states. Planning systems that use them are based primarily on the notion of state. In contrast, we contend that that a process-based analysis is more powerful and more expressive than analyses based on state-transitions. Indeed, we have already seen in the introduction that our process-based analysis is more powerful than state-based ones. Certain ideas cannot be expressed in a state-based analysis unless the agent has *a priori* knowledge of all possible states and their consequences. We turn briefly to the expressive advantages of our approach.

There are many actions that are not easily or naturally expressed in terms of transitions between states. For example, suppose there are several successful outcomes of an action: it is of course possible to build a disjunctive state description, such as "the state in which either the Daleks' ship has been marooned in deep space or it has been blown to smithereens". However, we believe that it is more natural to think in terms of the process

$$P = (Maroon \cdot \text{NIL}) \oplus (BlowUp \cdot \text{NIL})$$

than it is to reason about the transition from the state in which the ship is orbiting round our planet to the disjunctive state in which the ship has been somehow neutralized.

The notion of state-achievement is not sufficiently flexible to express all the goals that are necessary. The objective of an agent should *not* be to achieve a *given state* but rather to *attempt to perform an action*: if the Doctor yells at Jo, his assistant, to hit the red button, Jo's goal is not to achieve the state of the red button having been depressed or even that of the Tardis' having changed its location (the likely result of the action). Jo's goal is simply to attempt to perform the action of hitting the button, whether with her finger or by throwing something at it. Our approach, then, seems more expressive.

Returning to the power of our approach, consider the matter of *execution monitoring*. Armed with the supposedly foolproof plan

$$P' = GoToBridge \cdot DisableDavros \cdot$$

$$(WreckDrive \cdot \text{NIL} \oplus PlantBomb \cdot Escape \cdot \text{NIL}),$$

the Doctor and Jo set off for the Bridge. Unfortunately, it turns out that the Daleks are expecting trouble. Since they are on a difficult mission, they have placed twice as many guards in the corridors as had been anticipated in the plan P', and we are back to the situation discussed in the introduction. When the Doctor and Jo observe the guards, Jo reasons that they will be unable to make it past the first turn without being exterminated. The Doctor agrees with her. Thus they have recognized that the initial plan will be unworkable if they proceed around the corner. At this point, they have a choice: either they can give up on the attempt to neutralize the Daleks' ship, so abandoning our planet to its fate, or they can modify their plan on the basis of the new information (the existence of a duct).

This notion of execution monitoring provides opportunities for agents to identify (desirable) modifications to their plans. As

the Doctor runs along the corridor, for instance, he is also concurrently observing his environment. Seeing the air duct, and reasoning that he is unlikely to make it to the end of the corridor, he reasons that he should modify his plan and go along the duct.

This ability to interrupt an execution can be represented in algebraic terms. This representation relies on the combinator for *parallel composition of processes*: the process $P \parallel Q$ is one in which the process P and Q proceed at the same time, mutually interacting with each other and the environment. For example, if the Doctor is running and observing in parallel, or concurrently, his description as a process can be

$$Q = ((RunInBursts \cdot Q) + (ReachEnd \cdot \text{NIL})) \parallel R.$$

Note that this description is recursive. Interruptions of the recursive running occur as a result of observations, so the process R must allow the Doctor either to give up and try something else or to carry on running. In the definition

$$R = (NoOpportunity \cdot R) \oplus (SpotOpportunity \cdot X),$$

if there is no opportunity then the Doctor carries on running but if the Doctor spots an opportunity (a possible *synchronization* with the environment), he can perform the alternative process X. Here X is an unspecified process: the Doctor must determine by *reasoning* that he should go along the air-duct.

So far we have identified an agent with the plan it is attempting to execute. However, if the plan is to be modified on the basis of observation of the environment, *i.e.*, if the process X is to be determined, then the agent must be assumed to have a reasoning capability: this is treated in the next section.

Some of the advantages of process-based planning have already been noted by others. The Procedural Reasoning System (PRS) of Georgeff and Lansky [4, 5, 6, 8, 9], used in the SRI robot Flakey, was based on processes instead of state transitions. Lyons and Hendriks [12, 13] use processes as the basis of their representation language $\mathcal{RS}$ for their kitting robot. Lansky [10] introduces the notion of *action-based planning* for use in logistical planning domains. Firby's RAPS system [2, 3] and McDermott's Reactive Plan Language [15] are both essentially based on actions. We believe that the addition of a complete set of process-combinators and corresponding logics for reasoning about action and change will build on these successes.[4]

Reasoning and metareasoning

We conclude this paper with a discussion of reasoning about actions. There are two types of reasoning that we wish to facilitate: firstly, an agent must be able to reason about its own behaviour and that of its environment; secondly, we must be able, as designers of agents (and possibly of environments), to reason about the agent and its interaction with its environment.

Firstly, consider an agent. In order to be able to reason about its own behaviour, and hence construct and adapt plans, an agent needs a suitably *intensional*[5] logic that will allow it to derive processes that will achieve goals. Such a logic may be required to represent ideas such as hypothetical reasoning, probabilistic reasoning, belief, obligation and resource-sensitivity. For example, the Doctor's robotic dog K9 may build a plan to explore the ship on the basis that if it believes its batteries have 3 hours' supply left, then it be obliged to return to the entrance within $2\frac{1}{2}$ hours. Also, processes (plans) and propositions (descriptions of states, goals, *etc.*) can be *dependent on one another*. Designing such logics (there will be choices, depending on our philosophical and engineering stances) is a non-trivial task and is the focus of much of our current work.

Secondly, as designers, engineers and philosophers we want to reason about the agents that we build: we want to be able to *predict* how our agent will behave in given circumstances. In particular, we may want to make such statements as "our agent will

[4] Our theory also provides a powerful notion of how to compare and reason about plans.

[5] This is a rather delicate point: the agent may reason extensionally from its own point of view but intensionally from the point of view of an observer.

never use a plan that involves mass destruction", or "there is a plan that would work but our agent won't be able to find it". We need a logic in which we can express information about the processes that model agents.

Fortunately, process theory offers a natural logic for this purpose, *Hennessy-Milner* (HM) logic [16]. Classically, this is given by the following syntax:

$$\phi \quad ::= \quad V \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid [a]\phi \mid \langle a\rangle\phi,$$

where V ranges over basic propositions, $\wedge$ and $\vee$ are the usual classical conjunction and disjunction, and $[a]$ and $\langle a\rangle$ are *modalities* expressing the connection between propositions and change. Here truth is relativized, so that we are interested not in whether a formula always holds but in for which processes it holds:

$[a]\phi$ holds for a process if ϕ is true for all processes accessible via an a happening (so $[a]\phi$ expresses the necessity of ϕ holding after all a actions). Similarly,

$\langle a\rangle\phi$ expresses possibility: $\langle a\rangle\phi$ holds for a process if the process can do an a and then ϕ holds.

To see the logic in action, consider the process P'. We may be able to reason from P' that after any possible sequence of performing the actions *GoToBridge* then *DisableDavros* it will be possible to plant the bomb, since P' satisfies

$$[\mathit{GoToBridge}][\mathit{DisableDavros}]\langle \mathit{WreckDrive}\rangle \mathbf{true}.$$

This does not, of course, mean that the Doctor can draw the same inference: in performing this meta-reasoning we may be using information that is inaccessible to the agent.

HM logic turns out to provide a useful notion of equivalence — two processes are equivalent just in case exactly the same formulae are true of both of them. One of the first results of the rich metatheory of processes is that this logical notion of equivalence coincides with the behavioural notion of *bisimulation equivalence* [16]. Such an equivalence, then, identifies two plans just when they have the same outcomes in all settings.

Unfortunately, although HM logic allows us to describe and reason about possible behaviours in an elegant and intuitive way, it does not have enough structure to serve as the basis of the agent's own reasoning, with the requirements discussed above.

Current work is described in [17, 18, 19].

References

1. R. Fikes and N. Nilsson, *STRIPS: a new approach to the application of theorem proving to problem solving*, Artif. Intel. 2(1971), 189-208.
2. R.J. Firby, *Adaptive execution in complex dynamic worlds*, Technical Report YALEU/CSD/RR 672, Computer Science Department, Yale University, 1989.
3. R.J. Firby, *Task networks for controlling continuous processes*, in Proc. 2nd AIPS (K. Hammond, Ed.), 1994.
4. M. Georgeff, *Actions, processes and causality*, Reasoning about Actions and Plans: Proc. of the 1986 Workshop, Morgan Kaufmann, 1987.
5. M. Georgeff and A. Lansky, *Procedural knowledge*, Proc. of the Institute of Electrical and Electronics Engineers, 74(10) 1986, 1383-1398.
6. M. Georgeff, A. Lansky, and P. Bessiere, *A procedural logic*, in Proc. of the Ninth International Joint Conference on Artificial Intelligence, 1985.
7. M. Hennessy, *An algebraic theory of processes*, the M.I.T. Press, 1988.
8. A. Lansky, *Localized representation and planning methods for parallel domains*, Proc. of the Sixth National Conference on Artificial Intelligence, AAAI, 1987.
9. A. Lansky, *A representation of parallel activity based on events, structure, and causality*, Reasoning about Actions and Plans: Proc. of the 1986 Workshop, Morgan Kaufmann, 1987.
10. A. Lansky, *Action-based planning*, Proc. 2nd AIPS (K. Hammond, Ed.), 1994.
11. J.-M. Lofficier, *Dr. Who programme guide*, W.H. Allen, 1981.
12. D. Lyons, *Representing and analysing action plans as networks of concurrent processes*, IEEE Trans. on Robots and Automata, 9(3) 1993.
13. D. Lyons and A. Hendriks, *Testing incremental adaptation*, Proc. 2nd AIPS (K. Hammond, Ed.), 1994.
14. J. McCarthy and P. Hayes, *Some philosophical problems from the standpoint of artificial intelligence*, Machine Intelligence 4 (B. Meltzer and D. Michie, Eds.), Edinburgh Univ. Press, 1969.
15. D. McDermott, *A reactive plan language*, Technical Report YALEU/CSD/RR 864, Computer Science Department, Yale University, 1991.
16. R. Milner, *Communication and concurrency*, International series on computer science, Prentice Hall International, 1989.
17. D. Murphy, L. Pryor and D. Pym, *A process-theoretic perspective on the comparison of plans*, submitted, 1995.
18. L. Pryor, D. Pym and D. Murphy, *Processes for plan-execution*, submitted, 1995.
19. D. Pym, L. Pryor and D. Murphy, in prep., 1995.