A Note on Concurrency and Complexity in Temporal Planning

Maria Fox and Derek Long
Department of Computer & Information Sciences
University of Strathclyde, Glasgow, UK

Abstract

Rintanen recently reported (Rintanen 2007) that the complexity of temporal planning with durative actions of fixed durations in propositional domains depends on whether it is possible for multiple instances of the same action to execute concurrently. In this paper we explore the circumstances in which such a situation might arise and show that the issue is directly connected to previously established results for compilation of conditional effects in propositional planning.

1 Introduction

In his paper *Complexity of Concurrent Temporal Planning* (Rintanen 2007), Jussi Rintanen shows that temporal planning in propositional domains, with durative actions of fixed durations, can be encoded directly in a propositional planning framework by using (propositionally encoded) counters to capture the passage of time. Actions are split into their end points, in much the same way as shown in the semantics of PDDL2.1 (Fox & Long 2003) and as implemented in some planners (Halsey, Long, & Fox 2004; Long & Fox 2003). This encoding allows him to deduce that the complexity of this form of temporal planning is equivalent to that of classical planning when the number of such counters is polynomial in the size of the original (grounded) domain. However, if multiple instances of the same action may execute concurrently then it is not sufficient to have a single counter for each action instance, but instead as many counters are required as potential instances of the same action that may run concurrently. Rintanen observes that this could be exponential in the size of the domain encoding, placing the planning problem into a significantly worse complexity class than classical planning: EXPSPACE-hard instead of PSPACE-hard.

In this paper, we explore the situations in which instances of the same action can run concurrently and link the complexity costs the previously recognised problem of compiling conditional effects into classical propositional encodings (Gazen & Knoblock 1997; Nebel 2000).

2 Preliminaries

We begin by providing some definitions on which the remainder of the paper is based.

Definition 1 *A* classical propositional planning action, a, *is a triple,* $\langle P, A, D\rangle$, *where each of* P, A *and* D *is a set of atomic propositions. The action is applicable in a state,* S, *also represented by a set of atomic propositions, if* $P \subseteq S$. *The effect of execution of* a *will be to transform the state into the new state* $a(S) = (S - D) \cup A$.

Although states are sets of propositions, not all sets of propositions form valid states for a given domain. For a given domain, consisting of an initial state, a collection of actions and a goal condition, the set of states for the domain is the set of all sets of propositions that can be reached by legal applications of the actions. In the rest of the paper, when we quantify over states we intend this to be over all the valid states for the (implicit) domain in question.

Definition 2 *A* simple durative propositional action, D, *with fixed duration (Fox & Long 2003), is the 4-tuple* $\langle A_s, A_e, I, d\rangle$, *where* d *is the duration (a fixed rational),* A_s *and* A_e *are classical propositional planning actions that define the pre- and post-conditions at the start and end points of* D *respectively, and* I *is an invariant condition, which is a set of atomic propositions that must hold in every state throughout the execution of* D.

We do not choose to emphasise the conditions under which two classical actions are considered mutex, here (see (Fox & Long 2003) for details), but note that concurrent execution of two instances of the same durative action in which the end points coincide will not be possible if the end points are mutex. This means that they cannot delete or add the same propositions, so that they actually have no effects. Hence, there is no role for these actions in a plan and they can be ignored

in planning. Therefore, we assume that all our durative actions must, if two instances of the same action are to run concurrently, be executed with some offset between them.

3 Key Properties of Actions

We now proceed to define some essential properties that help to characterise the ways in which actions can interact with one another or with aspects of the states to which they are applied.

Definition 3 *A classical propositional action, $a = \langle P, A, D\rangle$ is* repeatable *if in every state S in which a is applicable, $P \subseteq a(S)$.*

A repeatable action can be applied twice in succession without any intervening action to reset the state of resources that might be used by the action. As we shall see, repeatable actions are constrained in the impact they may have on a state.

Definition 4 *A classical propositional action, $a = \langle P, A, D\rangle$ is* weakly conditional *if there are two states S_1 and S_2 such that a is applicable in both states and either there is a proposition $p \in A$ such that $p \in S_1$ and $p \notin S_2$ or there is a proposition $p \in D$ such that $p \in S_1$ and $p \notin S_2$.*

A weakly conditional action is one that can be executed in situations in which some of its positive effects are already true, despite not being preconditions of the action, or some of its negative effects are already false. The reason we call these actions weakly conditional is that these effects are semantically equivalent to the simple conditional effects *(when (not p) p)* and *(when p (not p))* for positive and negative effects respectively. These expressions are obviously part of a richer language than the classical propositional actions. In fact, they make use of both negative preconditions and conditional effects. This combination is known to be an expensive extension to the classical propositional framework (Gazen & Knoblock 1997; Nebel 2000). Nevertheless, weakly conditional actions are obviously valid examples of classical propositional actions. Notice that we require weakly conditional actions to be applicable in states that capture both possibilities in the implicit condition. This constraint ensures that situations in which the preconditions of an action imply that a deleted condition must also hold, without that condition being explicitly listed as a precondition (or the analogous case for an add effect) are not treated as examples of weakly conditional behaviour.

We now define some actions with reduced structural content of one form or another.

Definition 5 *A classical propositional action $a = \langle P, A, D\rangle$ is a* null action *if P, A and D are all empty.*

Definition 6 *A classical propositional action $a = \langle P, A, D\rangle$ is a* null-effect action *if for every state S such that $P \subseteq S$, $S = a(S)$.*

Note that one way in which an action can be a null-effect action is that the action simply reasserts any propositions it deletes and all of its effects are already true in the state to which it is applied. Actions that reassert conditions they delete are not entirely useless, provided they also achieve some other effects that are not already true. Some encodings of the blocks world domain can lead to ground actions that both delete and add the proposition that there is space on the table, simply to avoid having to write special actions to deal with the table. Also observe that null actions are a special case of null-effect actions.

We can now prove a useful property of repeatable actions:

Theorem 1 *Any repeatable action is either a weakly conditional action, a null action or a null-effect action.*

Proof: Suppose a repeatable action, $a = \langle P, A, D\rangle$, is not a null-effect action (and, therefore, not a null action). Then there must be some state in which a can be applied, S_a, such that $a(S_a) \neq S_a$. Since a is repeatable, it must be that $P \subseteq a(S_a) = (S_a - D) \cup A \neq S_a$.

Suppose a is not weakly conditional. Then for every $p \in D$, $p \in S_a$ iff $p \in a(S_s)$ and for every $p \in A$, $p \in S_a$ iff $p \in a(S_a)$. Since $A \subseteq a(S_a)$, the latter implies that $A \subseteq S_a$. The fact that $a(S_a) \neq S_a$ then implies that there is some $p \in D$ such that $p \in S_a$ and $p \notin a(S_a)$. This contradicts our assumption, so a must be weakly conditional. □

We now consider the ways in which these classical actions can appear in certain roles in durative actions.

Definition 7 *A simple durative action, $D = \langle A_s, A_e, I, d\rangle$, is a* pseudo-durative action *if A_e is a null action and I is empty.*

Definition 8 *A simple durative action, $D = \langle A_s, A_e, I, d\rangle$, is a* purely state-preserving action *if $A_e = \langle P, A, D\rangle$ is a null-effect action and every state satisfying I also satisfies P.*

3.1 Deadlocking Actions

One last variety of action is so significant we choose to devote a separate subsection to it.

Definition 9 *A simple durative action, $D = \langle A_s, A_e, I, d\rangle$, is a* deadlocking action *if there is a state, S, such that $I \subseteq S$ but A_e is not applicable in S.*

Thus, a deadlocking action is one that could begin execution and then, either by execution of intervening actions, or possibly simply by leaving

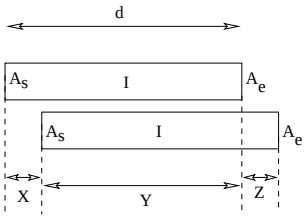

Figure 1: The intervals created by overlapping execution of two instances of the same action.

the state unchanged, it is possible to arrive in a situation in which the action cannot terminate because the conditions for its termination are not satisfied.

Deadlocking actions are clearly no natural actions: there is no real situation in which it is possible to stop time advancing by entering a state in which an action must terminate before time progresses, but cannot because the conditions for its termination are not satisfied. If we adopt a model in which a durative action has a fixed duration then the conditions for its termination must be inevitable, but the effects it has might well be conditional on the state at that time. In domains where deadlock is possible (for example, in the execution of parallel processes), the effect is not to stop time, of course, but to stop execution of the processes. This means that if one were to consider the behaviour of the parallel processes to be modelled by durative actions, the failure to terminate is handled by the actions having unlimited duration.

Therefore, we contend that no natural domains require to be modelled with deadlocking actions.

4 Self-Overlapping Actions

We now turn our attention to the circumstances in which two instances of the same simple durative action can be executed concurrently. Figure 1 shows the intervals that are created by the overlapping execution of such a pair of action instances. Note that when such an overlap occurs there are two places where classical propositional actions might be repeated: A_s and A_e.

Theorem 2 *If two instances of a simple durative action, $a = \langle A_s, A_e, I, d\rangle$ can execute concurrently, then either a is either a deadlocking, pseudo-durative or purely state-preserving action, or else A_e is weakly conditional.*

Proof: Suppose that two instances of a can execute concurrently and consider the two instances of A_e at the ends of the action instances. Either a is deadlocking, or else it must be possible for these two instances to be repeatable, since there is no requirement that an action be inserted in the period Z. Then, by our earlier result, A_e must be either a null action, a null-effect action or else weakly conditional. If A_e is a null action then a is either pseudo-durative (if I is empty) or else it is purely state-preserving. Finally, if a is not deadlocking and A_e is a null-effect action, then any preconditions of A_e must be true in any state satisfying I (otherwise there would be a state in which I was satisfied, yet a could not terminate, implying that a is deadlocking) and therefore a is either pseudo-durative (I is empty) or else it is purely state-preserving. □

Now that we have classified the simple durative actions that may execute concurrently with themselves, we briefly analyse the alternatives. We have already argued that deadlocking actions do not appear in natural domains. Pseudo-durative actions can be treated as though they were classical propositional actions, without duration, provided that a simple check is carried out on completed plans to ensure that adequate time is allowed for any instances of these actions to complete. Purely state-preserving actions are more interesting. An example of such an action is an action that interacts with a savings account that then triggers a constraint that the money in the account must be left untouched for some fixed period. Clearly, such an action is not unreasonable, even if it is uncommon. Fortunately, Rintanen's translation of temporal domains into classical domains can be achieved for purely state-preserving actions without additional counters to monitor the duration of overlapping instances of these actions. This is because the only important thing about these actions is how long the conditions they encapsulate must be preserved. Each time a new instance is executed, the clock must be restarted to ensure that the preservation period continues for the full length of the action from that point. Since the end of the action has no effects it is not necessary to apply it except when the counter reaches zero, at which point the invariant constraint becomes inactive.

Thus, the source of the complexity gap that Rintanen identifies can be traced, for all practical purposes, to the use of durative actions terminated by weakly conditional actions. Weakly conditional actions can be compiled into non-weakly conditional actions by the usual expedient of creating multiple versions of the actions. The idea is to have one version for the case where the condition is true and one for the case where the condition is false, each with the appropriate additional precondition to capture the case and the appropriate version carrying the conditional effect, but now as an unconditional effect. The problem with this compilation is that it causes an exponential number of variants to be created in the size of the collection of conditional effects.

In general, the current collection of benchmark domains do not appear to contain durative actions with repeatable terminating actions (although in many cases this is because the states in which the end actions can be executed are limited by the necessary application of the start effects of the durative actions to which they belong). This means that the problem of self-overlapping actions does not arise in these domains.

In domains in which there are repeatable terminating actions, it is non-trivial to identify which effects contribute to the weakly conditional behaviour. Delete effects are simpler to manage: any delete effect that is not listed as a precondition can be assumed to have the potential to be a weakly conditional effect. Add effects are more problematic: unless an add effect is shown to be mutually exclusive with the preconditions of the action, it must be assumed that it is weakly conditional. It is possible to use mutex inference, such as that used in Graphplan (Blum & Furst 1995) or that performed by TIM (Fox & Long 1998), to identify which add effects must be considered as weakly conditional. In general, to ensure that the weakly conditional behaviour has been completely compiled out, it is necessary to make a conservative assumption about any effects that cannot be shown to be ruled out. Nevertheless, in practical (propositional) domains the number of effects is tightly limited (ADL domains with quantified effects are not quite so amenable) and this makes it possible to compile out the weakly conditional effects with a limited expansion in the number of actions.

5 Relevance to Practical Planning

The relevance to practical planner design of the result we have demonstrated is two-fold. Firstly, we have shown that treatment of overlapping instances of the same action can only occur under limited conditions. These conditions can often be identified automatically using standard domain analysis techniques (Fox & Long 1998). This means that it is possible to determine whether machinery is required to be activated to handle the special case. Avoiding the use of techniques that would be redundant is useful in practical planner design, as a way to achieve improved efficiency.

Secondly, the results demonstrate that the focus of temporal planning should be, in the first place, on handling concurrency between distinct action instances and on the treatment of weakly conditional effects. The latter phenomenon is one that has not, to the best of our knowledge, been highlighted in the past, but is clearly a significant issue, since compilation of such effects into unconditional actions is both non-trivial and also, potentially, exponentially costly.

6 Conclusions

We have shown what kinds of simple durative actions can run concurrently with instances of themselves. Identifying the conditions that allow this has led to the discovery of a close link between the complexity gap identified by Rintanen and the complexity induced by the extension of propositional domains to those with conditional effects. A further important consequence of this analysis is to learn that if actions have bounded effect lists then the complexity of temporal planning is PSPACE-complete, even if self-overlapping actions are allowed.

References

Blum, A., and Furst, M. 1995. Fast planning through plan-graph analysis. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI95)*, 1636–1642.

Fox, M., and Long, D. 1998. The automatic inference of state invariants in TIM. *Journal of AI Research* 9.

Fox, M., and Long, D. 2003. PDDL2.1: An extension of PDDL for expressing temporal planning domains. *Journal of AI Research* 20:61–124.

Gazen, B., and Knoblock, C. 1997. Combining the expressivity of UCPOP with the efficiency of Graphplan. In *ECP-97*, 221–233.

Halsey, K.; Long, D.; and Fox, M. 2004. Crikey - a planner looking at the integration of scheduling and planning. In *Proceedings of the Workshop on Integration Scheduling Into Planning at 13th Internati onal Conference on Automated Planning and Scheduling (ICAPS'03)*, 46–52.

Long, D., and Fox, M. 2003. Exploiting a graphplan framework in temporal planning. In *Proceedings of ICAPS'03*.

Nebel, B. 2000. On the expressive power of planning formalisms: Conditional effects and boolean preconditions in the STRIPS formalism. In Minker, J., ed., *Logic-Based Artificial Intelligence*. Kluwer. 469–490.

Rintanen, J. 2007. Complexity of concurrent temporal planning. In *Proceedings of International Conference on Automated Planning and Scheduling*, 280–287.