Sequential vs. Non-sequential Temporal Planning Domains (Final project report)

Juraj Dzifcak
School of Computing and Informatics
Arizona State University
Tempe, AZ 85287-8809

Abstract

Temporal planning has seen many advances in the last several years, as shown by the temporal track in International Planning Competition. However, the analysis in (Cushing *et al.*, ICAPS 2007) shows that most of the domains used in the competition are inherently sequential and do not require concurrency. In addition, many of the winning planners did not really perform temporal planning and there was only a small and limited set of temporal languages they were capable of solving. In this work we take the presented analytical result, combine them with the work in (Fox & Long 2007) and implement some of these results and evaluate and analyze them on several planning domains.

Introduction

With the rise of the new PDDL language(2.1.3) (Fox & Long 2003) allowing to properly specify temporal planning domains, temporal planning has become more popular over last several years. In addition, the latest results encourage the idea that temporal planners seem to improve alongside classical planners suggesting a correlation in between those two. However, in (Cushing *et al.* ICAPS 2007) it is argued and shown that the domains used for the competition are inherently sequential, and thus do not require concurrency. In addition, one can look at the temporal languages as either temporally simple or temporally expressive (Cushing *et al.* IJCAI 2007), and that simple languages, such as TGP, are not capable to express many problems where the concurrency is required in order to solve them. Although more complex languages, such as PDDL allow to specify much larger set of problems, since they are superset of the simple languages it means that even when expressed in PDDL 2.1, the problem and the domain might not require concurrency at all. When combined with the fact that the well performing temporal planners are not complete for many temporal planning domains(In particular with the ones requiring concurrency), intuition might suggest that the actual competition problems do not require concurrency, as shown in (Cushing *et al.* ICAPS 2007). In terms of complexity, (Rintanen 2007) showed that, unless two instances of the same actions are allowed to execute concurrently, on can encode temporal planning in propositional domains with actions with fixed durations in propositional planning framework using counters as time points. And unless the number of those counters is exponential to the size of the grounded domain, the complexity level does not increase. However, allowing the instances of the same action to run concurrently, the number of counters becomes to high and increases the complexity of planning in that domain. In this work we continue above works and provide and implementation and analysis of to try to determine if a domain is inherently sequential or not.

Background

Let us now present some background information about the domains discussed in this work. The theorems and definitions are obtained from (Cushing *et al.* ICAPS 2007) and (Fox & Long 2007). Let us start with actions.

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 $a(S) = (S - D) \cup A$.*

Please note that we only assume a limited number of possible states. These are given by applying each applicable action to the initial state and repeating this process on all the resulting states. Let us now define a durative action.

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.*

Please note that this definition does not consider mutexes, however it is easy to verify that one cannot executed more than one instance of a durative action in case these several instances would form a mutex at the end points. Hence in this case it is safe to assume that if we want to execute 2 or more instances of a particular action at the same time, there must be an offset in between each of them.

Now that we have a representation for actions, we can define some properties of them.

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

Definition 4 *A classical propositional action, $\langle P, A, D \rangle$ 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 $p \in S1$ and $p \notin S2$ or there is a proposition $p \in D$ such that $p \in S1$ and $p \notin S2$.*

In other words, a weakly conditional action is an action that one can execute in states where one of it's positive or negative effects already hold. Let us now define some reductions in between the actions.

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

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

Intuitively, a null action is an empty action while null-effect action is an action that does not change the states in which it can be executed. Please note that this can be a rather complex action which adds and deletes the same propositions. Such action can actually be used to help in some domains, such as blocks world. Hence the following result can be given:

Theorem 1 *(Fox & Long 2007) Any repeatable action is either a weakly conditional action, a null action or a null effect action.*

Please note that checking this condition in general might be very hard, due to the fact that we have to be able to evaluate some of them over all the states. However, there are some specific cases n how to check this condition discussed later on.

The presented result does not talk about durative actions. However, we can extend the definitions and the result in a following way.

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 is a null-effect action and every state satisfying I also satisfies P.*

With this in mind, we can now present a stronger condition on repeating actions, deadlock. This is similar to deadlocks in many other problems, in other words we assume an action can start executing but during execution due to some interference(by either inside or outside) the action cannot finish and hence leads to a stop and deadlock. Let us now present a definition of deadlock for durative actions.

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.*

Please note that deadlocking actions do not really occur naturally. In real situations, there isn't really a way to stop the clock, so the action will still finish executing. However, it might not achieve the desired effects. Thus we can now formulate the following result:

Theorem 2 *(Fox & Long 2007)*
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.

Again, as in previous case, it might not be that easy to check for this condition for a temporal domain. As before, we will refer to one of the latter sections to discuss on how to check for these conditions.

In addition to the above discussion, the work by (Cushing *et al.* ICAPS 2007) discusses additional cases when a domain is not sequential.

Definition 10 *An action has temporal gap if*

1. *There is a condition or effect on a fluent x AT START*
2. *There is a condition or effect on a, possibly different, fluent y AT END*

A domain has temporal gap if any action in the domain has temporal gap. A domain forbids temporal gap if none of the domains actions has temporal gap.

Definition 11 *A problem has required concurrency if there exists a plan solving the problem and every such solution has concurrently executing actions. A domain has required concurrency if there exists any problem which has required concurrency.*

Please note that having timed exogenous effects and deadline goals in the domain definitions almost certainly leads to required concurrency.

Definition 12 *Two plans, which solve the same problem, are causally equivalent if they can be shown to be correct using the same causal-link proof.*

Following (Cushing *et al.*, ICAPS 2007), a causal-link proof consists of a set of causal-links (one for each goal and for all conditions of every action) and an ordering relation on effects and conditions; a causal-link matches each condition with a supporting effect. The ordering relation orders the supporting effect of a causal-link before its supported condition, orders action starts before their ends, and orders every threatening effect to a causal link either before the links supporting effect or after the supported condition (Chapman 1987). A plan can be shown to be correct using a causal-link proof if the dispatch times of the plans actions induces the ordering relation of the proof, and respects the actions durations.

Definition 13 *A plan, P, is inherently sequential if there exists a sequential plan P' solving the same problem, such that P and P' are causally equivalent. A problem is inherently sequential if every solution is inherently sequential. A domain is inherently sequential if every possible problem is inherently sequential. A language is inherently sequential if every expressible domain is inherently sequential.*

Please note that if a domain is inherently sequential, it does not have to require concurrency. Similarly, if a domain requires concurrency, it might still be inherently sequential. This is because there might be be multiple plans to solve the problem, some with and some without concurrency. However, it seems that such domains seem to be quite rare.

Using the properties described above, one can formulate the following theorem:

Theorem 3 *If a domain forbids temporal gap, the domain is inherently sequential.*

In addition, we can introduce the notion of causal independence in between actions to obtain a more stronger condition of sequentiality.

Definition 14 *An action A is AT START-causally independent if there cannot be a direct causal ordering between the beginning of A and some concurrent action:*

1. *For all AT START conditions x, there is no possibly concurrent action B with an effect y threatening x*
2. *For all AT START effects x, there is no possibly concurrent action B with either a condition y supported by x nor an effect y conflicting with x.*

Definition 15 *An action A is AT END-causally independent if there cannot be a direct causal ordering between the ending of A and some concurrent action:*

1. *For all AT END conditions x, there is no possibly concurrent action B with an effect y supporting x*
2. *For all AT END effects x, there is no possibly concurrent action B with an effect y conflicting with x.*

Theorem 4 *If every action in the domain is AT START causally independent or AT END causally independent then the domain is inherently sequential.*

It is interesting to also include the following result(please note that in order for it to hold, some domains have to be 'repaired')

Theorem 5 *Every temporal domain in IPC 2002, 2004, 2006 is inherently sequential.*

Initial ideas

As a brief sidestep, let us review the initial ideas of this project. Initially, the idea was to focus on re planning. Later on, this was narrowed to trying to come up with a progression planner for temporal planning domains which would be able to somehow realize he 'missed' an important time point and get back and re plan. While this idea seems interesting and definitely has some merit, after further discussion it was concluded that even if an efficient method of determining when to return and to what position in time to return(which is still questionable at this point), this kind of planner could not compete well with current partial order planners. This is because in general and on average, it does require more overhead and the expected payoff is not as high. Hence it was concluded that while such planner may be effective in some domains, overall it would be really hard to outperforms current planners, and hence this idea was not continued. Instead, evaluation of temporal planning domains was performed and implemented. Also a theoretical result about the null effect actions was investigated, but it was concluded that it is only a slight modification of the definition and theorem presented above.

Implementation

Let us now discuss the implementation aspects of this work. First, please note that the goal was not to come up with a full decision procedure, due to complexity reason. Deciding if a domain does require concurrency might be as hard, if not harder, as actual planning. Instead, the goals was to design and implement several methods that can help in deciding this, or, using the provided theoretical results, can confirm the domain/problem does not require concurrency. Since as said, these are not mutually exclusive, a certain domain may fir into both categories.

Let us now discuss some aspects of the results presented in previous section. Theorem 1 and theorem 2 specify properties of action which allow for concurrency. The problem, is just by looking at the definitions, checking for these properties seems to be too hard and complex since we have to check for all the states, which is extremely slow and inefficient. However, taking a closer look one can make following simple observations

Observation 1 *If an action has no effects, it is a null-effect action. Furthermore, if it'a add and delete effects are the same,it is a null-effect action.*

Hence we can use these to make a partial decision procedure and for all such actions, we can safely assume they can be executed concurrently. In addition, we implemented a method to check the for deadlock and some additional properties based on a particular state, which might be useful for other applications such as planners.

Similarly, let us take a closer look at theorem 3 and theorem 4. In this case the property of temporal gap and causal independence can be checked in quadratic time with respect to the number of actions in a straight forward way. However, even in these case one can observe that the causal independence property is reflexive, and hence can can only check each pair of actions only once, halving the time required to preform this check.

In the future we plan to include more optimizations to further improve the efficiency of the checkers.

Also, we assume conditional effects are precompiled into a domain without such effects using one of the available tools.

We can divide the general algorithm into following steps:

1. Parse the domain
2. Check for temporal gaps and print out the actions with them. If no gaps are found, the domain must be inherently sequential.
3. Check for At Start and At End causal independence. If every action is found to be independent, the domain must be inherently sequential.
4. Check for action concurrency. If an action cannot occur concurrently with itself report this action.

5. Based on the above points, classify the domain as either inherently sequential or unknown.

Please note that the implementation cannot decide if a domain does require concurrency at the moment.

Domains

For the purposes of testing we will use a set of 'classical' domains from the competition. Since we know these are inherently sequential, we will add some more domains based on light and match scenario(see for example (Cushing *et al.*, ICAPS 2007)) where the concurrency is required. While the time to evaluate this domains was taken, the results over several runs were not conclusive enough to include it in this presentation.

Evaluation

Let us now present the results of the implementation.

Test no	Domain	Sequential?	Time
1	Depots	Yes	N/C
2	DriverLog	Yes	N/C
3	ZenoTravel	Yes	N/C
4	Light and match 1	Unknown	N/C
5	Light and match 2	Unknown	N/C

In all cases, the running time varied to much and hence is not conclusive (N/C). However, overall the running time was very fast, under 0.1 secs. For first three domains, it was concluded they are indeed inherently sequential, while for the rest the methods did not provide enough conclusive evidence to categorize these. (Although these were designed so that they do require it.)

Conclusions and discussion

In this work we presented and discussed various methods in determining if a temporal planning domain does or does not require concurrency. Several of the proposed methods were implemented and evaluated on some of the interesting planning domains. We hope that this can be a first step toward having an off line/on line temporal domain evaluator. The discussed framework can furthermore be extended as need as new results about concurrency in temporal planning arrive. Furthermore, the approach presented here cannot fully decided the domains, and is only capable of producing the 'Yes' answer or the 'Unknown' answer. A method to be able to say 'No' for some domains could be very useful in many scenarios.

Acknowledgments

I would like to thank W. Cushing, J. Benton and S. Kambhampati for their help and guidance in this work.

Bibliography

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32(3):333377.
Cushing, W.; Kambhampati, S.; Weld, D. ; Mausam. 2007. When is temporal planning really temporal?. *IJCAI*
Cushing, W.; Kambhampati, S.; Talamadupula, K.; Weld, D. ; Mausam. 2007. Evaluating temporal planning domains. *ICAPS*
Fox, M. and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR*
Fox, M. and Long, D. 2007. A note on concurrency and complexity in temporal planning. *PLANSIG*
Rintanen, J. 2007. Complexity of concurrent temporal planning. *ICAPS*