ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1698

Delete-Free Reachability Analysis for Temporal and Hierarchical Planning

Arthur Bit-Monnot[1] and **David E. Smith**[2] and **Minh Do**[2]

Abstract. Reachability analysis is a crucial part of the heuristic computation for many state of the art classical and temporal planners. In this paper, we study the difficulty that arises in assessing the reachability of actions in planning problems containing sets of interdependent actions, notably including problems with required concurrency as well as hierarchical planning problems. We show the limitation of state-of-the-art techniques and propose a new method suitable for both temporal and hierarchical planning problems. Our proposal is evaluated on FAPE, a constraint-based temporal planner.[3]

1 Introduction

Reachability analysis is crucial in computing heuristics guiding many classical and temporal planners. This is typically done by relaxing the action delete lists and constructing the reachability graph. This graph is then used as a basis to extract a relaxed plan, which serves as a non-admissible heuristic estimate of the actual plan reaching the goals from the current state.

Temporal planning poses some additional challenges for reachability analysis as heuristics should not only estimate the total cost but also the earliest time at which goals can be achieved. This can be accomplished on the reachability graph by labeling: (1) propositions with the minimum time of the effects that can achieve them; and (2) actions with the maximum time of the propositions they require as conditions. Since the reachability graph construction process progresses as time increases, when all *start* conditions are reachable, a given action a is eligible to be added to the graph. However, there is the additional problem that a's end conditions must also be reachable, although they do not need to be reachable until the end time of a. To see why this is a problem for the conventional way of building the reachability graph, consider the two actions in Figure 1: action B achieves the end condition for action A, but requires a start effect of A before it can start. Thus, B cannot start before A, but A cannot end until after B has ended. This means that A is not fully reachable until B is reachable, but B is not reachable unless A is reachable. Whether this turns out to be possible depends on whether B fits inside of A. In this example, the reasoning is simple enough, but more generally, B might be a complex chain of actions.

Planners such as POPF [2] address this problem by splitting durative actions into instantaneous start and end events, and forcing a time delay between the start and end events. In our example, the start of A would be reachable, leading to the start of B being reachable, which leads to the end of B being reachable, and finally the end of A being reachable. This approach therefore concludes that A is reachable. Unfortunately, the same conclusion is reached even when

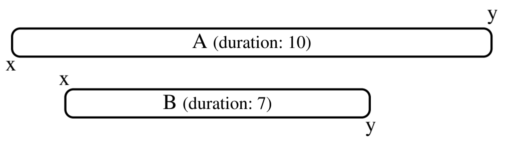

Figure 1: Two interdependent actions: A with a start effect x and an end condition y, and B with a start condition x and an end effect y.

B does not fit inside of A, because this "action-splitting" approach allows A to "stretch" beyond its actual duration.

In this paper, we present an approach to reachability analysis that addresses the above limitation and show how it can be beneficial for both generative and hierarchical temporal planners.

2 Planning Model & Relaxation

Temporal Model. We consider temporal planning problems similar to those of PDDL 2.1. For ease of presentation, we consider a discrete time model and restrict ourselves to actions with fixed duration and positive conditions.[4] An action has a set of conditions C_a and a set of effects E_a, all at arbitrary instants in the action envelope. A condition $c \in C_a$ has the form $\langle [t_c]\, f_c \rangle$, where f_c is a fluent and t_c is a positive delay from the start of the action to the moment f_c is required to be true. An effect $e \in E_a$ has the form $\langle [t_e]\, f_e \rangle$ (resp. $\langle [t_e]\, \neg f_e \rangle$ for negative effects), where t_e is the positive delay from the start of the action to the moment the fluent f_e becomes true (resp. false).

Relaxed Model. To estimate when each fact can be achieved, our reachability analysis utilizes *elementary actions*, which are artificial actions created from the original temporal actions defined in the domain description. Elementary actions contain: *(i)* only a single 'add' effect and *(ii)* the minimal set of conditions required to achieve that effect. More specifically for each positive effect $e = \langle [t_e]\, f_e \rangle$ of an action a, there will be an elementary action a_e with:

- a single effect $\langle [1]\, f_e \rangle$
- for each condition $\langle [t_c]\, f_c \rangle$ of a, a condition $\langle [t_c - t_e + 1]\, f_c \rangle$

Our relaxed model is composed of those delete-free elementary actions, each giving one possible way of achieving a given fluent. For any given elementary action, we say that a condition $c = \langle [t_c]\, f_c \rangle$ is a *before-condition* (resp. an *after-condition*) if $t_c \leq 0$ (resp. $t_c > 0$). An after-condition represents a condition that is required when or after the effect of the elementary action becomes active (e.g. y is an after-condition of the action A of Figure 1). In a reachability graph, such conditions would be represented by a negative edge. Those after-conditions are necessary for the presence of interdependencies such as the one shown in Figure 1 [3].

[1] LAAS-CNRS, Université de Toulouse, email: arthur.bit-monnot@laas.fr
[2] NASA Ames Research Center, email: {david.smith, minh.do}@nasa.gov
[3] A long version of this paper was presented at the HSDIP workshop [1].
[4] Extensions to more general models are discussed in the long version of this paper [1].

3 Reachability Analysis with After-conditions

In POPF, the splitting mechanism used for reachability analysis results in ignoring all after-conditions of durative actions. In order to avoid this additional relaxation, our method for reachability analysis is based on repeatedly alternating two steps: *(i)* we optimistically propagate achievements times while ignoring after-conditions; then *(ii)* we enforce all after-conditions. More specifically:

1. As a preliminary, we select a set of symbols that are assumed reachable at time 0. All fluents of the initial state are obviously part of this set. They are optimistically complemented with all elementary actions that have no before-conditions.
2. We then iteratively extend the set of assumed reachable nodes with: *(i)* all fluents that have an assumed reachable achiever and *(ii)* all actions whose every before-condition is assumed reachable. Each reachable symbol is associated with an earliest appearance time satisfying: *(i)* the minimal delays between an action and its before-conditions and *(ii)* the minimal delay between a fluent and its first achiever. This is done by a Dijkstra-like procedure that processes the nodes by increasing their earliest appearance times.
3. Our optimistic assumptions are then revised *recursively* by incorporating the ignored after-conditions. Specifically: *(i)* any elementary action with an after-condition on an unreachable fluent is removed from the model, *(ii)* if a removed action a is the only achiever of a fluent f, f is removed together with any action depending on it, *(iii)* the minimal delays between an action and its after-conditions are enforced by increasing the action's earliest appearance time as much as necessary.
4. If any action was updated in the previous step, we go back to step (2) and restart the propagation of earliest appearances from the updated nodes. Otherwise, analysis finishes with a set of reachable actions and fluents, each associated with an earliest appearance time.

Because earliest appearances could be endlessly increased towards infinity, we complement the last step with a detection of unreachable nodes. A group of nodes $\mathcal{N}$ is unreachable if for any node $n \in \mathcal{N}$, there is a delay of at least d_{max} between the earliest appearance of any node $n' \notin \mathcal{N}$ and n, d_{max} being the highest delay between any condition and its action or any action and its effect. The intuition is that nodes of this group are delaying each other due to unachievable interdependencies [1].

4 Extension to Hiearchical Planning

While automated reachability analysis is widespread in generative planners, hierarchical planners still rely on manual annotation of methods for this purpose. Here we propose a translation of hierarchical actions that exposes hierarchical features as additional conditions and effects for the purpose of reachability analysis.

We associate each hierarchical action a to a task symbol τ_a and a set of subtasks S_a. The intuition is that a achieves the task τ_a and requires all its subtasks to be achieved by other actions. For a plan π to be a solution it is required that:

- all initial tasks and all subtasks have been *achieved*. A task τ spanning a duration $[st_\tau, et_\tau]$ is said to be achieved if there is an action a_τ in the plan that: *(i)* achieves the task τ; *(ii)* starts at st_τ; and *(iii)* ends at et_τ.
- all actions in π achieve some task. This simulates HTN planners, in which all actions are inserted to achieve a pending task.

To allow reasoning on those additional requirements, we transform a hierarchical action a, with task τ_a, into a 'flat' action a_{flat} with:

- all conditions and effects of a,
- one start condition $\langle [0]\, required(\tau_\mathrm{a}) \rangle$,
- one start effect $\langle [0]\, started(\tau_\mathrm{a}) \rangle$ and one end effect $\langle [d_a]\, ended(\tau_\mathrm{a}) \rangle$, where d_a is the duration of a,
- for each subtask $\langle [d_1, d_2]\, \tau \rangle$ of a:
  - two conditions $\langle [d_1]\, started(\tau) \rangle$ and $\langle [d_2]\, ended(\tau) \rangle$,
  - one effect $\langle [d_1]\, required(\tau) \rangle$.

Actions resulting from this compilation step encompass both causal and hierarchical features of the domain and can be split into elementary actions for reachability analysis techniques described earlier. This transformation usually exposes many interdependencies as each action both enables and requires the presence of its subactions.

5 Experiments & Conclusion

Our technique has been implemented in FAPE [4], a constraint-based temporal planner for the ANML language supporting both hierarchical and generative planning. Reachability analysis is used to *(i)* prune the search space by detecting dead-ends; and *(ii)* disregard resolvers involving unreachable actions.

Our method is tested with different configurations, R_∞ being the original method and R_5 and R_1 are variants in which the number of iterations are limited to 5 and 1 respectively. R^+ is the configuration where all after-conditions are ignored, thus producing the same result as the reachability analysis of POPF. $\varnothing$ denotes the configuration in which no reachability analysis is performed. Evaluation was done on various temporal domain with and without hierarchies.

	R_∞	R_5	R_1	R^+	$\varnothing$
(IPC-8) satellite (20)	14	14	14	14	**15**
(IPC-5) rovers (40)	**25**	**25**	**25**	**25**	**25**
(IPC-2) logistics (28)	**8**	**8**	**8**	**8**	**8**
(IPC-8) satellite-hier (20)	**17**	**17**	**17**	**17**	16
(IPC-5) rovers-hier (40)	**22**	**22**	**22**	**22**	**22**
(IPC-8) tms-hier (20)	**7**	**7**	**7**	**7**	**7**
(IPC-2) logistics-hier (28)	**28**	**28**	**28**	6	9
(LAAS) handover-hier (20)	**16**	**16**	**16**	7	7
(IPC-8) hiking-hier (20)	**20**	17	16	15	17
(LAAS) docks-hier (18)	**17**	13	12	7	7
Total (254)	**174**	167	165	128	133

Table 1: Number of solved tasks for various domains with a 30 minutes timeout. The best result is shown in bold. The number of problem instances is given in parenthesis.

As shown in Table 1, our method results in significant performance gain on hierarchical problems. This is because those problems feature many examples of interdependent actions for which our method is especially usefull. On temporally simple problems (here non-hierarchical ones), our method is equivalent to the reachability analysis of POPF and does not result in any runtime penalty. Note that the use of reachability analysis is here limited to dead-end detection. While this proves extremely useful on a wide variety of problems, one could also contemplate using it as a base for heuristic extraction.

REFERENCES

[1] Arthur Bit-Monnot, David E. Smith, and Minh Do, 'Delete-free Reachability Analysis for Temporal and Hierarchical Planning', in *Heuristics and Search for Domain-independent Planning*, pp. 93–102, (2016).

[2] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long, 'Forward-Chaining Partial-Order Planning', in *ICAPS*, pp. 42–49, (2010).

[3] Martin C. Cooper, Frederic Maris, and Pierre Régnier, 'Managing Temporal Cycles in Planning Problems Requiring Concurrency', *Computational Intelligence*, **29**, 111–128, (2013).

[4] Filip Dvorak, Roman Bartak, Arthur Bit-Monnot, Felix Ingrand, and Malik Ghallab, 'Planning and Acting with Temporal and Hierarchical Decomposition Models', in *ICTAI*, pp. 115–121, (2014).