Using a Resource-Based Framework to Integrate Planning and Replanning

Roman van der Krogt
Delft University of Technology
P.O.Box 5031, 2600 GA Delft
The Netherlands
R.P.J.vanderKrogt@ITS.TUDelft.NL

30th March 2003

1 Introduction

Many contemporary planning systems (such as those that have taken part in the International Planning Competition [10]) assume that planning is a type of problem in which each instance that is to be solved is a static and stand-alone problem. That is, (i) the assumption is made that planning can be solved *off-line* and (ii) these planners have no history. However, when we look at real life planning problems, we see the following characteristics: (i) the planning problem is part of a *dynamic environment*, which means that planning should be considered an *on-line* problem rather than an off-line one and (ii) planners are used to solve problems from domain that they have already seen. Another difference between real-world problems and most planning systems is that in reality, planning is usually not performed by a single, autocratic entity: resources are often controlled by other parties, and some goals may only achieved with the help of other agents.

Of course, there are systems that at least partially deal with these problems. For example, case-based planners such as SPA [7] and CHEF [6] use a history of past solutions to find an appropriate starting plan to begin planning with. This plan is then modified to match the current goals and initial state. Other systems focus on the replanning aspect to take into account the dynamic nature of planning. For example, GPG [5] can be used to repair a plan when a discrepancy is discovered between the original planning problem and the current state of the world. The term *continual planning* [3] is used to refer to systems in which planning, replanning and execution is continuously interleaved. A variety of techniques is used to tackle multi-agent domains. For example *plan-merging* approaches, such as [4, 11], or *distributed planning* approaches such as [14, 2].

Our goal is to bring together ideas from case-based planning, multi-agent planning and replanning into a unifying framework that enables agents to reason about their plans and their relations with other agents. One of the benefits of such a framework would be that it should help transfer existing single-agent algorithms to a multi-agent context. Also it may help in finding new heuristics and algorithms that are more efficient in real world domains.

The remainder of this paper is structured as follows: first we briefly describe the ARF framework [1] that will be used to describe planning problems and solutions. Then we show how planning and replanning can both be conceived as *plan transformation* problems. These problems can be solved by applying a certain set of plan operators, that make use of a plan library to store knowledge from previous problems. A reformulation of Kambhampati's Refinement Planning framework [9] can be used to solve these problems. Finally, we sketch how the refinement planning framework could be extended to include multi-agent planning. We end this paper by presenting some ideas for future work.

2 The Action Resource Formalism

This section briefly describes the Action Resource Formalism (ARF). For a more detailed description, see [11, 1].

In propositional domain models, usually each attribute of an object in the world relevant to the planner is described by a separate proposition. The main idea behind the ARF is that these propositions can be merged into a single description of the object: this is called a *resource* fact (or just resource). Such a resource describes the *type* of the object as well as the *sorts* and the *values* of its *attributes*.

A state is described by enumerating all the resources that are true at a certain point in time. *Actions* describe state transformations. An action is a rule that states which resources are removed from the state, and which resources are added to it. A plan is a partially ordered set of actions, along with a description of which resource produced by an action is consumed by another action. A plan P is *adequate* for an initial state I and a goal state G when P can be executed from state I and produces goals that satisfy G. The tuple (I, G) where I is an initial state and G a goal state is called a *context*. When A a plan P is adequate for (I, G), we call the tuple (I, P, G) adequate.

3 Planning and Replanning

We can treat both planning and replanning as variations of the same problem. Both problems have the same characteristics: (i) an adequate tuple (I, P, G) is available, (ii) an inadequate tuple (I', P', G') exists and (iii) the goal is an adequate tuple (I', P'', G').

In a *planning* domain (i) signifies a plan that the planner knows (by experience) a plan P that is adequate for a context (I, G).[1] The planner proposes (ii) as a starting point, using P for P' and (iii) has to adapt P' to derive a plan P''

[1]This may be the empty plan $\emptyset\!\!\!/$ being adequate for the empty context $(\emptyset, \emptyset)$.

that is adequate to the current context. In a *replanning* context, (i) describes that there used to be an adequate plan P, but (ii) due to changes in the initial resources, the goals state or the available actions, the (possibly changed) plan P' is no longer adequate for the (new) context (I', G'), but (iii) has to be transformed to a plan P'' that is adequate for this new context. Thus: both planning and replanning can be described by a plan transformation process. We propose two plan operators to be used in this process: the addition operator $\oplus$ and the deletion operator $\ominus$.

The addition operator $\oplus$ glues two plans together, similar to action concatenation in regular planning systems. This operator is given the two plans that are to be concatenated and a description of how this concatenation has to take place. It can be used to satisfy a goal by adding a plan that produces this goal.

Sometimes, however, we are interested in a resource that is used by the plan to produce a resource in which we are not interested. For this case, we introduce the deletion operator $\ominus$: given a plan and a subplan, it removes the subplan, thus freeing up the resources that this subplan requires. This operator can also be used to preprocess a plan to be able to glue it to some other plan.

To be able to use the plan operators, we need to be able to retrieve plans that may be used during the transformation. This storage area is called the *plan library*. This library can be conceived as a simple collection of plans from which the planner may choose. Assuming (i) that for each action in the domain there is at least one plan in the library that uses this action, and (ii) that the internal structure of each of the plans in the library is known, it can be shown that these operators form a *complete* set of operators, i.e. given a proper set of plans, an adequate plan (if existing) can always be created. Even more so, if the adequate plan P requires actions from k plans to be combined, we need at most $2k - 1$ applications of the operators to construct the plan.

4 Refinement Planning

We can modify the Refinement Planning Approach [9] to make it applicable to the ARF such that it makes use of the plan operators and plan library described above. By doing this, we obtain a template algorithm for planning and replanning. However, this requires that we drop the constraint that each refinement step should *reduce* the set of possible refinements. After all, during a replanning step, we may have to retract a part of the plan, thus *enlarging* the set of possible refinements. Therefore, we have to ensure that the refinements made during such steps have to be chosen such that the same partial plan is not generated twice, in order to prevent a possible infinite regression.

As a proof of concept, the FF-algorithm [8] has been implemented on top of the ARF [12]. In [13] we show how the SPA [7] system for plan adaptation can be seen as an instance of this template. We also show how a multi-agent planning technique can be incorporated by extending the modified refinement planning template.

5 Conclusions and Future Work

The current work has shown what kind tools we need to integrate planning and replanning approaches. The ARF can be used to model plans and problems, and the plan operators can be used to modify these plans. Plans can be stored in the plan library, and the refinement planning approach gives us a template algorithm for finding the right sequence of plan transformations. There are still vital points missing, points that we hope to resolve in the future.

To begin with, we are pursuing the adaptation of an existing approach to include planning using a plan library, to assess the impact of using such a library on the planning performance (both in time as well as in quality). The second subject that we are looking at is how a plan library can created and maintained. A third topic that we plan to pursue is to find an algorithm based on the refinement planning template that integrates solving planning and replanning problems.

References

[1] Mathijs M. de Weerdt, André Bos, J.F.M. Tonino, and Cees Witteveen. A resource logic for multi-agent plan merging. *Annals of Mathematics and Artificial Intelligence, special issue on Computational Logic on Multi-Agent Systems*, 37(1–2):93–130, January 2003.

[2] Mathijs M. de Weerdt and Roman P.J. van der Krogt. A method to integrate planning and coordination. In M. Brenner and M. desJardins, editors, *Planning with and for Multiagent Systems*, number WS-02-12 in AAAI Technical Report, pages 83–88, Menlo Park, CA, 2002. AAAI Press.

[3] Marie E. DesJardins, Edmund H. Durfee, Charles L. Ortiz, and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 4:13–22, 2000.

[4] Eithan Ephrati and Jeffrey S. Rosenschein. Multi-agent planning as the process of merging distributed sub-plans. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence (DAI-93)*, pages 115–129, May 1993.

[5] A. Gerevini and I. Serina. Fast plan adaptation through planning graphs: Local and systematic search techniques. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (AIPS-00)*, pages 112–121, 2000.

[6] Kristian J. Hammond. Case-based planning: A framework for planning from experience. *Cognitive Science*, 14(3):385–443, 1990.

[7] S. Hanks and D.S. Weld. A domain-independent algorithm for plan adaptation. *Journal of AI Research*, 2:319–360, 1995.

[8] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of AI Research*, 14:253–302, 2001.

[9] Subbarao Kambhampati. Refinement planning as a unifying framework for plan synthesis. *AI Magazine*, 18(2):67–97, 1997.

[10] Derek Long and Maria Fox. International planning competition. http://www.dur.ac.uk/d.p.long/competition.html, 2002.

[11] J.F.M. Tonino, André Bos, Mathijs M. de Weerdt, and Cees Witteveen. Plan coordination by revision in collective agent-based systems. *Artificial Intelligence*, 142(2):121–145, 2002.

[12] Roman P.J. van der Krogt, Mathijs M. de Weerdt, Leon R. Planken, and Ard Biesheuvel. Cabs planner. http://www.pds.twi.tudelft.nl/~mathijs/, 2003.

[13] Roman P.J. van der Krogt, Mathijs M. de Weerdt, and Cees Witteveen. Integrating planning and replanning, manuscript. http://www.pds.twi.tudelft.nl/~roman/, 2003.

[14] William E. Walsh and Michael P. Wellman. A market protocol for decentralized task allocation and scheduling with hierarchical dependencies. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS-98)*, pages 325–332, 1999. An extended version of this paper is also available.