Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Failure Handling In a Planning Framework

Sertac Karapinar and **Sanem Sariel-Talay**
Artificial Intelligence and Robotics Laboratory
Computer Engineering Department
Istanbul Technical University, Istanbul, Turkey
{karapinarse,sariel}@itu.edu.tr

Introduction

When the real outcomes of actions are not completely represented, a planner may not be able to construct a valid plan even if there exists one. This research focuses on generic domain update and reasoning methods to construct alternative plans against real-time execution failures which are detected either during runtime or earlier by a plan simulation process. Based on the updated domain representations, a new executable plan is constructed even when the outcomes of existing operators are not completely known in advance or valid plans are not possible with the existing representation of the domain.

TLPlan, a forward chaining temporal planner originally proposed by Bacchus and Ady (2001), is used as the temporal planner in this system since it can construct plans for durative and concurrent multirobot actions. A search node in TLPlan includes the world state and the action applied along with the world clock that defines the start time of that action. Applying an action at a state means that this action is scheduled for the world clock of that state. The search proceeds by applying new actions or advancing the world clock toward finding a complete temporal plan. When the agent executes its plan, the execution may be ceased due to unforeseen cases. Under these circumstances, the planner may not be able to construct a valid plan. In order to enable the agent construct a new plan, failures should be handled by investigating the causes of failures if reasoning tools are available.

Some of the earlier studies have investigated action execution failures in planning frameworks. Goebelbecker et al. (2010) deal with a problem that is quite similar to our problem. Their purpose is finding explanations for failures by generating excuses, and making the agent replan with the help of these excuses. Excuses are extracted by changing the initial state such that a valid plan can be constructed from there on.

Beetz et al. (2010) propose a system to cope with plan execution flaws. Their system involves plan projection and transformation processes in order to detect behavior flaws and generate a new plan. Plan projection is achieved by simulating the plan execution in a physics-based simulator. In case of a behavior flaw in the plan projection, transformation rules are applied to find a valid plan.

Our approach differs from earlier work in that failures are handled by an integrated reasoning, planning and learning approach. We use PROBCOG to extract the cause of failure and extend the knowledge base (KB) to cope with different types of failures. If the reasoner cannot help in recovering from the failure, an intuitive approach is applied to construct alternative plans.

In a previous study (Usug and Sariel-Talay 2011), a failure handling system has been developed that does not use reasoning tools. In case of a failure, the agent considers all actions that can manipulate the cause of the failure. This means that the agent tries to perform each of these actions until it handles the failures and succeeds in achieving its goals. This might lead the agent to execute irrelevant actions, causing degradation in performance. For example, let's assume the agent is tasked to move an object from its initial location to a desired final destination. When the agent detects an obstacle on its way and there is not an alternative path, the only option for the agent is taking alternative actions that may resolve the issue. The previous approach can achieve this by finding actions that may change the state of the obstacle. If the domain involves both paint or push actions (i.e., corresponding operators), both of them could be selected since both of these actions may directly affect the cause of the failure, obstacle. To make a difference between paint and push actions, domain knowledge about utilities of these actions for the given circumstances and reasoning tools are needed.

Problem Statement

During the execution of a plan, an agent may face several failure situations. After detecting a failure, the agent should recover from that failure. Considering their recovery processes, the failure types can be grouped under two categories, namely, *temporary* and *permanent* failures. A temporary failure occurs during the execution of a plan. These types of failures can be resolved by replanning. However, there is no way to resolve a permanent failure. The problem that we investigate asks for finding the reasons of failures during the execution of a plan and recovering from them if they are *temporary* failures. There may be one or more actions that can surpass the cause of the failure; however, the

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

agent may not be aware of them. When the effects of these types of actions do not directly contribute to the ultimate goal of the agent, the planner may fail to add these actions to the constructed plan. A solution to this problem should successfully add those actions into the plan so that failures are resolved. It is obvious that a reasoning mechanism is needed to realize the necessity of these types of actions to achieve the goal. Therefore, integrating a reasoner is a crucial part of our work to ensure a more systematic way to handle failures.

A Case Study

As a motivating example to illustrate the above mentioned problem, a scenario can be given with a robot and several objects to be moved from their starting positions to the given goal positions. The robot is able to execute several actions such as *pick up*, *put down* objects by its gripper and *move* from one location to another. However, the robot might drop an object while trying to pick it up. This failure might occur because of several reasons such as the weight of the object or wrong grasp position. The robot can handle the failure by executing pick up action with different parameters if the reason of the failure is its wrong grasp position. If the robot fails in achieving picking up the object because of its weight, there is no way to pick it up. Replanning may lead the robot keep dropping the object repeatedly. Other the other hand, another action (e.g., push) which does not directly contribute to the robot's goal can handle this type of failure. To resolve the failure, this action should be added into the plan. A reasoning mechanism can figure out the cause of the failure in such cases and updates the KB so that the necessary actions are included in the plan.

Reasoning on Failures

We combine the TLPlan planner with a first-order probabilistic reasoner PROBCOG to solve the issue mentioned above. TLPlan is used to construct a temporal plan. If the plan fails during its execution, PROBCOG uses the KB and the incoming information from the agent's sensors, and makes new conclusions about the cause of the failure. Two types of conclusions can be made: (1) the parameters of a failed action are updated to handle failures, (2) a permanent failure is detected when the robot gives up executing an action that fails after several trials. The KB is also extended to include these new conclusions that enables replanning. The updated planning problem is fed into the planner to re-plan for the given situation.

The reasoning tool enhances the planning process in three ways:

- New conclusions can be made about the world when the agent is failed to execute an action.
- New conclusions can be made to find an executable action that might resolve the failure.
- Further conclusions about the utility of future behaviors can be made considering specific properties of objects.

The following example from our example scenario shows how our reasoning module works. The robot tries to pick up a red cubical object. When it fails in its first attempt, the parameters of pick up action are updated so that the robot changes its orientation w.r.t the object and tries to pick up the object again. If it fails in its second attempt, the orientation is updated again. At the end of the third attempt, the KB includes the following facts:

$$Red(x)\Lambda Cube(x)\Lambda Orientation(\alpha, x)\Lambda PickUpFailed(x)$$
$$Red(x)\Lambda Cube(x)\Lambda Orientation(\beta, x)\Lambda PickUpFailed(x)$$
$$Red(x)\Lambda Cube(x)\Lambda Orientation(\gamma, x)\Lambda PickUpFailed(x)$$

If the above facts are the only knowledge in the robot's KB, a permanent failure assumption is made and the following hypothesis is constructed:

$$[Red(x)\ \Lambda\ Cube(x)] \Rightarrow PickUpFailed(x)$$

After trying to pick up some other red cubical objects and making new observations, the hypothesis might be strengthened or rejected.

Conclusion

In this paper, we propose combining a planner with a reasoner to help an agent handling failures during its plan execution. This is achieved even when the planner cannot come up with a valid plan considering the domain operators due to lack of exact information on the actual outcomes of actions that may resolve the failure. We show that if a reasoning tool is not available to the agent, the problem could be resolved at the expense of irrelevant actions. The intuitive idea that is applied in this case is finding actions that may change the state of the failure. If a reasoner, PROBCOG in our case, is provided, then more efficient solutions can be attained. Our ongoing work includes a method to find actions that have direct relations to the cause of failure. By using this approach, irrelevant actions are not selected anymore.

References

Bacchus, F., and Winter, M. A. 2001. Planning with resources and concurrency a forward chaining approach.

Beetz, M.; Jain, D.; Msenlechner, L.; and Tenorth, M. 2010. Towards performing everyday manipulation activities. *Robotics and Autonomous Systems* 58(9):1085 – 1095.

Gobelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up with good excuses: What to do when no plan can be found.

ProbCog. 2012. Probcog tool box. http://wwwbeetz.informatik.tu-muenchen.de/probcog-wiki/index.php/Main˙Page.

Usug, U., and Sariel-Talay, S. 2011. Dynamic temporal planning for multirobot systems.