Plan recovery in reactive HTNs using symbolic planning

Lydia OULD OUALI
LIMSI-CNRS, UPR 3251, Orsay, France
Univ. Paris-Sud, Orsay, France
Email: ouldouali@limsi.fr

Charles RICH
Worcester Polytechnic Institute
Worcester, MA, USA
Email: rich@wpi.edu/

Nicolas SABOURET
LIMSI-CNRS, UPR 3251, Orsay, France
Univ. Paris-Sud, Orsay, France
Email: Nicolas.Sabouret@limsi.fr

Abstract—

I. Introduction

Automatic planning is an important field of controlling artificial agents in complex and dynamic environments where research built two different approaches. The first one is symbolic planning: this approach consists in constructing a complete symbolic and logical model of the environment that allows the agent to reason about this model and define a complete plan to carry out its goals. The most popular architecture used to describe the environment is the hierarchical architecture HTN (Hierarchical Taks Network) [Ero96], which allows a recursive decomposition of complex goals into sub-goals or primitive actions. The HTN architecture eases the design of the environment and gives more expressiveness. Symbolic planning assumes that the environment is fully defined. By consequence, the agent is able to predict all the possible situations to plan in advance. Nevertheless, it becomes clear that authoring a complete representation of a dynamic and complex environment such as simulation of human behavior [CGS98] or the definition of dialog systems [AF02] requires significant knowledge-engineering effort [ZHH$^+$09], and even reveals to be impossible [Mae90]. However, with incomplete knowledge the agent cannot anticipate the future and the generated plan might be not executed as expected. Therefore, if at any point of the execution the plan breaksdown (i.e action execution fails), the planner has to stop the execution and build another plan that achieves the agent's goals. Such operation might be costly in terms of time and resources.

Because of these limitations, another planning approach called reactive planning was proposed [Fir87]. Reactive planning avoids long-term prediction in order to make the execution faster. For this reason, it leaves all the planning during the execution phase: the agent plans only for the next step to be executed from the current defined state of the environment. Thus, it can adapt the next step according to the observed changes. The main advantage of reactive planning systems is they don't need a complete definition of the environment. Instead, they aim to define the policy of the agent in its environment by running through a pres-authored HTN structure with procedural knowledge. Procedural knowledge defines conditions in the HTN domain knowledge as black-box procedures (for exmaple : JavaScript code) that contains no logical information (i.e no symbolic knowledge). This type of reactive HTN eases the design, reduce the complexity of planning and still can cope with complex dynamic environments [Bro05]. They are used in numerous application domains, such as dialog systems [BR03] and simulating human behavior [Bro05].

Nevertheless, breakdowns can still appear in reactive planning. An action execution can fail and leads the HTH to a state where no action can be applicable to achieve the goal. In such situation, the agent has to stop and think about a new solution to reach its goal. However, without symbolic knowledge, the agent has nothing to reason about. The execution thus stops and the agent cannot recover from its breakdown.

In order to deal with this limitation, we propose in this paper to extend reactive HTNs with a linear symbolic planner. For this reason, we propose to the HTN author to extend the procedurale konwledge of the HTN with some symbolic knowledge that allows the symbolic planner to compute local recovery plans. We study the capacity of such model to recover from breakdowns in reactive planning.

In section 2, we briefly present existing works in this domain. In section 3 we formalize the proposed solution *Discolog* and describe its implementation. Section 4, presents the expriments and discusses the obtained solutions. At the end, we discuss the futures works to validate and extends our solution to differents domains and uses.

II. Backgorund and related works

The proposed work aims to present a contribution to repair breakdowns in reactive HTNs. Thus, we propose a hybrid planning system that combines a reactive HTN with a basic linear symbolic planning system. In this section, we present the definition of the two systems involved and the existing approaches related to ouw work.

A. Hierachical and linear symbolic planning

linear planning system attempts to generate a plan to reach a goal state i.e. a sequence of actions such that starting from an initial state, the plan leads to the goal state. STRIPS planner [FN72] is the first contribution based on the linear planning methods. STRIPS is based on mean-end planner relied on a simple backward chaining search in the state space.

Another contribution in planning is the Hierarchical planning approach called HTN (Hierarchical Task Network)

[Ero96]. HTN was created as extension of the linear planner STRIPS allowing the planner to add more information and expressibility to the domain knowledge. HTNs can be represented as AND/OR tree where AND nodes represents the tasks and OR nodes defines the recipes of tasks.

- Primitives tasks represented as leaf nodes in the tree, are similar to linear planning actions and can be directly executed in the world. In addition, HTN tasks have post-conditions that specify when a task execution successes or fails i.e(a task execution is considered as complete when its post-conditions are satisfied).
- Compound tasks involve several tasks and can be performed by decomposing them into a sequence of subtasks using a specific recipe.
- Recipe represents a method to achieve or decompose a compound task. Each recipe is defined with an applicability condition that helps the planner choosing the appropriate decomposition for a task if there is more than one.

The HTN planning systems entents to plan for one or more goal task. Planning proceeds using task decomposition that starts from the initial goal task , decomposes it using a corresponding recipe, and breaksdown the goal into sequence of simpler subtasks. This process is applied recursively until until a conflict-free plan can be found. the plan consists on sequence of fully ordered primitive tasks that can make the goal task successful.

The HTN planners becomes popular this last decade and in different domains where several sys- tems were developed these recent years such as SHOP [NCLMA99], SIPE [Wil88] or NOAH [Sac75].

B. Related works and previous approachs

1) Reactive planning: Reactive planning becomes very popular in AI such in controlling mobile agents [BBC$^+$05], simulating human behaviour [BS01] or Gaming with the name f behaviour trees. Behaviours trees have the same hiearchical structure of HTNs and do Real time decision which can be seen as reactive planning. Nevertheless, As reactive planning is used for highly dynamic environements it presents certain limits as discussed below:

C. Brom [Bro05] proposes in his work an educational toolkit for prototyping human-like behaviour. the proposed reative architecture was based on the work proposed in [BS01]. Nevertheless, this reactive planning system faces some limits such as : impossibility to add new goals during the execution or inhebit an undesirable subtask.unatural switching between behaviour.

R. James Firby [Fir87]

2) Plan repair: plan repair by extending the generated plan with graph containing the causal links between the HTN's tasks [AKYG07] [WHLUMA07] [VDKDW05] [BD02] plan repair using heurestic [HTHO06]

III. MOTIVATION EXAMPLE

HTN domain knowledge:

Tasks

Load (Object X, Place Y, Place Z) : Precond: isOn(X,Y) . Postcond : isOn(X,Z).
Unload (Object X, Place Y, Place Z) : Precond: isOn(X,Y) . Postcond : isOn(X,Z).
Move (Object X, Place Y) : Precond: isOn(X,Y) . Postcond : ! isOn(X,Y).
PutDown(Object X,Place Y) : Precond: ! isOn(X,Y) . Postcond : isOn(X,Y).

Primitive tasks

Hold(Object X, RobotArm Arm) : Precond: ! isOn(X,Arm) . Postcond : isOn(X,Arm).
RaiseUp(Object X, Floor) : Precond: isOn(X,Floor) . Postcond : !isOn(X,Floor).
Release(Object X, RobotArm arm) : Precond: isOn(X,Arm) . Postcond : ! isOn(X,Arm).
put(Object X, Place Y): Precond: !isOn(X,Y) . Postcond : isOn(X,Y).

Recipes

holdwithOneArm(X) : Applicability cond: X.weight $<$ 5 Kg
ReleasewithOneArm(X) : Applicability cond: X.weight $>$ 5 Kg and weight $<$ 10 Kg

IV. SOLUTION

In this section we present our proposition based on extending the reactive HTN with symbolic planner to recover from breakdown.

Reactive HTNs do tasks decomposition and plan only for the next primitive task to execute from the current state, using for that a procedural domain knowledge. This later represents the policy of the agent in its environment as a set of procedures which can only be executed. Nevertheless, as the domain knowledge is never complete breakdowns might happens.Thus, the HTN can no longer proceed with the execution. In order to recover from such breakdown, the agent has to think of a strategy, but without symbolic knowledge and planner the agent has no way to think.

To overcome this problem, we propose to extend the reactive HTN with a symbolic linear planner that uses symbolic knowledge extracted from the HTN.

Some procedures in the procedural knowledge have the form of boolean procedures. this later can thought as predicate. For example, the *isOn(X,Y)* procedure return either true or false depending on the object X is on the place Y. This procedure can be represented as a predicate that takes two variables. Therefor, We propose to the HTN designer to covert theses procedures to a symbolic predicates and the primitives actions containing theses predicates.

Thus, if a breakdown occurs (The agent has nothing to execute and the goal is not reached yet). The hybrid system calls the recover procedure describe in the algorithm 1. It first traverse the HTN to detects all the tasks in the HTN affected by the breakdown *(FindCandidate procedure).* The affected tasks by the breakdown have one of their pre/post conditions which are no longer supported in the current state. Once all the task

Algorithm 1 Reactive planning and plan recovery algorithm

```
 1: procedure HYBRID SYSTEM(DomainKnowlege,Goal)
 2:     $\pi \leftarrow Reactive_HTN(DomainKnowlege,Goal)$
 3:     if $\pi \leftarrow$ Success then
 4:         return Success
 5:     else
 6:         $plan \leftarrow Recover(Goal)$
 7:         if (plan = null) then
 8:             return Failure
 9:         else
10:             for each action $a_i \in$ plan do
11:                 $Discolog(HTN, a_i)$
12:             end for
13:         end if
14:     end if
15: end procedure
16: procedure RECOVER(Goal)
17:     Candidates $\leftarrow$ findCandidate$(G)$
18:     if Candidates = $\emptyset$ then
19:         return null
20:     else
21:         $\Pi \leftarrow \emptyset$
22:         for each candidate $\in$ Candidates do
23:             $\Pi$ += LinearPlanner(candidate, CurrentState)
24:             Cost $\leftarrow \{cost(\pi) | \pi \in \Pi\}$
25:         end for
26:     end if
27:     return $\pi \in \Pi$ with minimum cost($\pi$)
28: end procedure
29: procedure FINDCANDIDATES(Goal)
30:     for each child $\in$ Goal do
31:         if status (child) = failed then
32:             add condition(child) to Candidates
33:         end if
34:         findCandidates$(children(child))$
35:     end for
36:     return Candidates
37: end procedure
```

candidates are detected, the linear planner is called for each candidate. The linear planner uses the symbolic knowledge to construct a plan that takes as goal repairing the failed condition in the task candidate. Finally, the hybrid system calculates the best and most promising plan to be passed to the reactive HTN in order to be executed.

A. Implementation of the solution

In this section we present the implementation of our solution called Discolog. Discolog is based on reactive HTN called Disco [**?**] and a simple STRIPS planner implemented in Prolog. We choose the Prolog implementation because of its inference engine that allow the planner reasoning on the possible links between the predicates.

Disco uses the ANSI/CEA-2018 standard for the procedural definition of its domain knowledge. Thus, Tasks are modeled using XML format. Primitive tasks contains grounding script parameter defined as JavaScript program which represent the effect of the primitive task execution in the environment.

Like reactive HTN, disco has no logical knowledge and plans only for the next step. Each task has a status which represents its state in the HTN:

- if the preconditions of the task are hold in the current state, the status of this task becomes *live* else the status of the task is *notDone* .
- Once the execution of a task is terminated, Disco evaluates its postconditions, if their are valid then the task execution is successful and its status becomes *Done*. Nevertheless, if task's postconditions are not valid, then the status of the task becomes *Failed*.

The status of the tasks are used to calculate the tasks candidates for the recovery procedure as described in the algorithm IV-A

For example, taking back the example of the robot, Both recipes to decompose the load task are not applicable which make the execution breaks down. Discolog calculates all the tasks and recipes affected by this breakdown by checking. In this case, all the remaining tasks to decompose the load task are taken as candidates. the planner is then called for each one. Using the symbolic knowledge the planner proposes as plan to decompose the object into two objects and load them separately as represented in the figure (attacher une figure de la reparation de plan).

```
procedure FINDCANDIDATES(Goal)
    for each child $\in$ Goal do
        if $(precondition(child)! = \emptyset$ and status (child)$\notin$\{Done, Live, Blocked\}) then
            add precondition(child) to Candidates
        else if (postcondition(child)!=$\emptyset$ and status (child)$\in$ \{Failed\} then
            add postcondition(child) to Candidates
        end if
        if (status $\in$ \{Live\} and nonPrimitive(child) and applicability(child)!= $\emptyset$) then
            add Applicability-condition(child) to Candidates
        end if
        findCandidates$(children(child))$
    end forreturn Candidates
end procedure
```

V. Experiments and results

1. Approach of the expriments: Test the capability of Discolog to recover from a breakdown given a certain amount of symbolic knowlege.

2. Benshmark creation:

Random HTNs with synthetic data. Breakdown caused in each primiive task. The purpose is to study the abbility of

Discolog to find a plan recovery for all possible breakdowns in the HTN. Symbolic data generation : the variation of the level of symbolic knowledge to insert in the linear planner domain knowledge

3. Present the obtained results and discuss them.

results obtaind of tree (5,4,1) (2,3,2) (3,3,3)

Discuss the fact that the more symbolic knowledge we have the more recovery we get. Expose the fact that we can not have a 100 of symbolic knowelge and its is limited to the representation of the HTN hauthor which is also incomplete.

VI. Conclusion

Remind the context of our work. the proposition and its adventages. the future work :

1. present system support for authoring reactive HTNs.
2. dialog system using Discolog

References

[AF02] James Allen and George Ferguson. Human-machine collaborative planning. In *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space*, pages 27–29, 2002.

[AKYG07] N Fazil Ayan, Ugur Kuter, Fusun Yaman, and Robert P Goldman. Hotride: Hierarchical ordered task replanning in dynamic environments. In *Planning and Plan Execution for Real-World Systems–Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop. Providence, RI*, 2007.

[BBC+05] Eric Beaudry, Yannick Brosseau, Carle Côté, Clément Raïevsky, Dominic Létourneau, Froduald Kabanza, and François Michaud. Reactive planning in a motivated behavioral architecture. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1242. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[BD02] Guido Boella and Rossana Damiano. A replanning algorithm for a reactive agent architecture. In *Artificial Intelligence: Methodology, Systems, and Applications*, pages 183–192. Springer, 2002.

[BR03] Dan Bohus and Alexander I Rudnicky. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. 2003.

[Bro05] Cyril Brom. Hierarchical reactive planning: Where is its limit. *Proceedings of MNAS: Modelling Natural Action Selection. Edinburgh, Scotland*, 2005.

[BS01] Joanna J Bryson and Lynn Andrea Stein. Modularity and design in reactive intelligence. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 1115–1120. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[CGS98] Rosaria Conte, Nigel Gilbert, and Jaime Simão Sichman. Mas and social simulation: A suitable commitment. In *Multi-agent systems and agent-based simulation*, pages 1–9. Springer, 1998.

[Ero96] Kutluhan Erol. Hierarchical task network planning: formalization, analysis, and implementation. 1996.

[Fir87] R James Firby. An investigation into reactive planning in complex domains. In *AAAI*, volume 87, pages 202–206, 1987.

[FN72] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[HTHO06] Hisashi Hayashi, Seiji Tokura, Tetsuo Hasegawa, and Fumio Ozaki. Dynagent: An incremental forward-chaining htn planning agent in dynamic domains. In *Declarative Agent Languages and Technologies III*, pages 171–187. Springer, 2006.

[Mae90] Pattie Maes. *Designing autonomous agents: Theory and practice from biology to engineering and back*. MIT press, 1990.

[NCLMA99] Dana Nau, Yue Cao, Amnon Lotem, and Hector Muñoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2*, pages 968–973. Morgan Kaufmann Publishers Inc., 1999.

[Sac75] Earl D Sacerdoti. A structure for plans and behavior. Technical report, DTIC Document, 1975.

[VDKDW05] Roman Van Der Krogt and Mathijs De Weerdt. Plan repair as an extension of planning. In *ICAPS*, volume 5, pages 161–170, 2005.

[WHLUMA07] Ian Warfield, Chad Hogg, Stephen Lee-Urban, and Héctor Munoz-Avila. Adaptation of hierarchical task network plans. In *FLAIRS Conference*, pages 429–434, 2007.

[Wil88] David E Wilkins. *Practical planning: Extending the classical AI planning paradigm*. Morgan Kaufmann Publishers Inc., 1988.

[ZHH+09] Hankz Hankui Zhuo, Derek Hao Hu, Chad Hogg, Qiang Yang, and Hector Munoz-Avila. Learning htn method preconditions and action models from partial observations. In *IJCAI*, pages 1804–1810, 2009.