Model and Graphical Tool to Formalize Human-Robot Interaction Based on Automated Planning

Alba Gragera[1], **Ángel García-Olaya**[1], **Fernando Fernández**[1], **Rebeca Marfil**[2]
[1] Universidad Carlos III de Madrid, Spain
[2] Universidad de Málaga, Spain
agragera@pa.uc3m.es, {agolaya, ffernand}@inf.uc3m.es, rebeca@uma.es

Abstract

The implementation of use cases in Social Autonomous Robotics is a complex and time-consuming task to be developed by domain experts and engineers, involving a large knowledge acquisition process. The resulting use case description must also be formalized taking into account stochastic events that may occur in the real world. Existing works rely on Automated Planning to deploy robotic use cases, where the standard Planning Domain Description Language (PDDL) is assumed. In order to facilitate to domain experts the description of the use case we propose a novel tool to create the model through state transition diagrams. From this diagram, the system automatically generates the PDDL files. A video demonstration is available[1].

Introduction

Social Autonomous Robotics working in real scenarios must act according to the environment and show flexible and robust behaviors, useful in dynamic and changing situations. Some approaches in the literature (Bandera et al. 2016; Cashmore et al. 2015; González, Pulido, and Fernández 2017; Mohseni-Kabir, Veloso, and Likhachev 2020) rely on Automated Planning (AP) to achieve this autonomous behavior by using a problem solver and a control architecture: the problem solver creates a plan of actions to be performed and the control architecture deals with execution and sensing. However, problem solving in realistic scenarios requires a large amount of knowledge to be processed by specific reasoning engines. It involves both knowledge on the application field and expertise in programming, in the frame of a time-consuming knowledge engineering process, presenting a bottleneck for developers and an entry barrier for novice users. Therefore, a growing demand for frameworks helping users to develop models in a seamless manner has emerged in the last years. Although there are tools to model AP domains prior to this work (Vaquero et al. 2013; Simpson, Kitchin, and McCluskey 2007), they usually require deep knowledge about the specification language and become unmanageable for large domains, in addition to the lack of features for interactive scenarios.

[1]http://bit.ly/icaps2021demo

In this work we propose a knowledge representation based on Classical Planning concepts but suitable for people with little knowledge of its formal language. Thank to this, users can define their own use case using a simple language based on a workflow representation of the expected robot behaviour. Once the model is created it can be automatically translated into its PDDL formalization. It provides a functional model ready to be injected into an AP-based control architecture, where the uncertainty is solved by replanning (Yoon, Fern, and Givan 2007).

AP Knowledge Representation Model

Knowledge engineers have to recognise the information involved in the desired use case to model it according to the AP concepts. Represented as predicates, it is essential at runtime to distinguish which information is dynamic (can change during execution), sensed from the exterior (objects colours, etc.), internal, or static or persistent (data that should not be removed, such as room locations). Such information is grouped into states, which conform the current knowledge the agent must reason with. Since reasoning usually involves just some of the facts present in the state, a partial state definition is enough to reason about it. To transit between successive states the user also has to define actions. A simple way to elicit them is to ask the expert what characteristics the scenario must hold to carry out the action (represented by an state) and how that state changes after its execution, transiting to another new state. Therefore, both states are connected by the action, where effects must be specified by the user to indicate how the action changes the world. These components shapes an agent-based model through a workflow representation. Finally, the user has to specify goals to set the real purpose of the robot.

Planning Tasks as Workflows

Based on the previous concepts, the agent model is depicted through a directed graph. It can be formalized using the concept of *options* (Sutton, Precup, and Singh 1999), composed by a state where the option starts, the policy followed throughout the option, and the partial state where the option terminates. Figure 1 shows a complex Social Robotics use case where the different options that a robot and a child can carry out are shown, such as the initial phase to introduce the exercise or or the placement of the blocks to put them in a certain order. Is the task of the reasoning engine to choose the most promising options sequence to solve the use case.

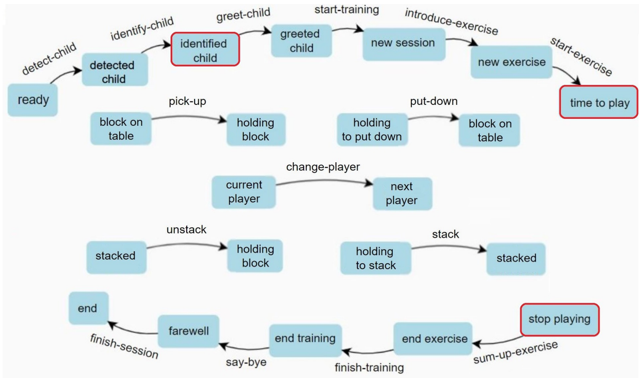

Figure 1: Use case involving an interactive blocks game

To be able to combine different options we also define *partial options*: non-empty sequences of states and actions which may be incomplete, being open to interleave any number of actions to meet intermediate states. Figure 2 considers a real use case in which relatives of residents in a retirement home can choose a time slot to make a video call by using a robot with a tablet entering the room. The option *at nursery - out of the room* represent the general video call process, which needs to interleave another options such us being disinfected or navigate to reach specific locations. Together with corrective actions to solve unexpected situations, users can also choose checkpoints from which to recover again the normal execution (highlighted in red in Figure 1).

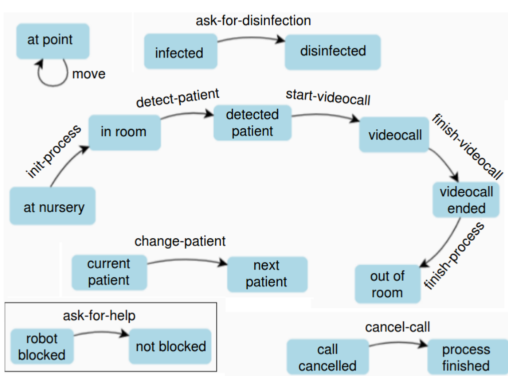

Figure 2: Videocall robot use case

From Workflows to PDDL

The model of the use case is automatically translated into its PDDL formalization, taking into account the next concepts:

Sequential actions add the state from which they start as precondition of the action, while effects are taken from the action definition. Parameters are defined by the objects involved in the facts of the preconditions and effects.

Loops are represented by two actions leaving from the same state. Such state must contain two possible situations for the exit condition: it is different from the current value of the control variable (assigned with the action that stays inside the loop), or it is equal to it (added as precondition of the action that leaves the loop).

Recovery actions are tasks defined by the expert to deal with unexpected situations (*ask-for-help* action in Figure 2). They will be included by replanning only after such cases.

Checkpoints work as restoration points, defined by the expert as places where the execution is desired to return after correcting an exogenous situation, instead of the point where the interruption took place.

Backwards actions are used for this *checkpoint* recovery, deleting all intermediate effects (excepts the information marked as persistent or sensed) added between successive checkpoints, forcing to restart the execution from the desired point. The compiler automatically generates one of these actions for each checkpoint.

Conclusions

We implemented a tool for Social Robotics use cases development, facilitating its modelling and formalization by using a workflow representation. Such model is automatically translated into its corresponding PDDL code. An interesting future line is to improve it with controls for flagging inconsistent domain construction or incompleteness in the model, warning the user about possible loss of information.

References

Bandera, A.; Bandera, J. P.; Bustos, P.; Calderita, L. V.; et al. 2016. CLARC: a Robotic Architecture for Comprehensive Geriatric Assessment. In *WAF*, 1–8.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; ; et al. 2015. ROSPlan: Planning in the Robot Operating System. In *ICAPS*, 333–341.

González, J. C.; Pulido, J. C.; and Fernández, F. 2017. A three-layer planning architecture for the autonomous control of rehabilitation therapies based on social robots. *Cognitive Systems Research (CSR)* 43: 232–249. ISSN 1389-0417.

Mohseni-Kabir, A.; Veloso, M.; and Likhachev, M. 2020. Efficient Robot Planning for Achieving Multiple Independent Partially Observable Tasks That Evolve over Time. In *Proceedings of the ICAPS*, 202–211.

Simpson, R. M.; Kitchin, D. E.; and McCluskey, T. L. 2007. Planning domain definition using GIPO. *Knowledge Eng. Review* (2): 117–134.

Sutton, R. S.; Precup, D.; and Singh, S. P. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.* 112(1-2): 181–211.

Vaquero, T. S.; Silva, J. R.; Tonidandel, F.; and Beck, J. C. 2013. itSIMPLE: towards an integrated design system for real planning applications. *Knowledge Eng. Review* 28(2): 215–230.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In Boddy, M. S.; Fox, M.; and Thiébaux, S., eds., *ICAPS*, 352. AAAI.