J.C. González, F. Fernández, A. García-Olaya, R. Fuentetaja. (2017). On the Application of Classical Planning to Real Social Robotic Tasks. In PlanRob 2017 Proceedings of the 5th Workshop on Planning and Robotics (ICAPS 2017)(pp. 38-47)

On the Application of Classical Planning to Real Social Robotic Tasks

José Carlos González, Fernando Fernández, Ángel García-Olaya and Raquel Fuentetaja
Planning and Learning Group, Department of Computer Science
Universidad Carlos III de Madrid, Leganés 28911 - Madrid, Spain
{josgonza, ffernand, agolaya, rfuentet}@inf.uc3m.es

Abstract

Automated Planning is now a mature area offering several techniques and search heuristics extremely useful to solve problems in realistic domains. However, its application to real and dynamic environments as Social Robotics requires much work focused, not only in the efficiency of the planners, but also in tractable task modeling and efficient execution and monitoring of the plan into the robotic control architecture. This paper identifies the main issues that must be taken into account while using classical Automated Planning for the control of a social robot and contributes some practical solutions to overcome such inherent difficulties. Some of them are the discrimination between predicates for internal control and external sensing, the concept of predicted nominal behavior with corrective actions or plans, the continuous monitoring of the plan execution and the handling of action interruptions. This manuscript highlights the dependencies between all the design and deployment activities involved: task modeling, plan generation, and action execution and monitoring. A task of Comprehensive Geriatric Assessment (CGA) is used as an illustrative example that can be easily generalized to any other interactive task.

Introduction

Simple or very specialized robots can work just with reactive behaviors, but autonomous robots that must take decisions and change their behavior according to the context, as social robots do, must have a deliberation process at some point. Automated Planning (Ghallab, Nau, and Traverso 2004) is very useful to manage this deliberation and, in particular, classical planning is able to exploit many of the latest contributions and advancements of the field. Classical planning already has impressive results finding plans very fast in multiple domains, but it still has a low deployment in real life applications. This manuscript tries to bring classical Automated Planning techniques closer to real social robotic scenarios.

Social robots (Leite, Martinho, and Paiva 2013) has to deal autonomously with people, requiring the monitoring of highly dynamic environments with a lot of uncertainty. Therefore, executing a plan of actions with real social robots is not straightforward because an action can fail or maybe the plan can become unfeasible due to some change in the environment. Joining planning and execution is not deeply studied in the literature although the need has already been pointed out (Ghallab, Nau, and Traverso 2014).

The execution must be controlled to change the plan depending on the evolution of the environment. Interruptions in the execution of the plan can interfere with a coherent social interaction, so the modeling of the domain has to take this matter into account. This environment can be composed of low-level information from the sensors that needs to be abstracted to high-level data.

In this work we apply classical planning interleaved with execution. Classical planning has several advantages over other automated planning models as hierarchical or probabilistic. It allows much more compatibility with existing planners which can be used as black boxes. Also, the newest heuristics are made for classical planning, so we can take advantage of them. Uncertainty is not represented explicitly, but it can be controlled with existing planning architectures based on replanning techniques. Regarding the representation language we use PDDL (Edelkamp and Hoffmann 2004; Fox and Long 2003), the standard language developed by the academic community. Specifically, our model requires types, negative preconditions and numeric fluents. PDDL facilitates the modeling of the knowledge in our domain.

The main objective of this manuscript is to highlight the issues that arise when joining classical planning and execution into a social robot.

Challenges of Social Robotics

Social Robotics is focused on all those robots that must interact socially with humans. Achieving a natural and fluent interaction is currently a huge challenge and an active research topic (Tapus, Matarić, and Scasselati 2007). Human communication includes many verbal and non-verbal elements and it is in the act of talking when this interaction becomes more sophisticated. A social robot will be useful only if it can interact socially and efficiently, so a social robot should have many of the characteristics of conversational agents.

In general, every social interaction involves some sort of transmission of information that can be driven by the structure of a conversation. If this structure is altered, then the communicative act could be incoherent or difficult to follow. The characteristics that a natural communication should have are also studied by fields as Pragmatics (Warren 2006)

and, depending on the objective, it has different phases that must be performed in order.

Apart from microphones and speakers to allow verbal communication, social robots can have non-verbal communication mechanisms as faces shown in a screen, robotic faces, lights, etc. Elements as touchscreens can also be used to circumvent current limitations of audio and image recognition, for instance. A mechanism to coordinate all these elements to achieve the most coherent and natural interaction as possible is essential in Social Robotics.

Additionally, these robots must function in very dynamic environments. In other words, they must respond coherently to a great amount of unexpected situations which can cause interruptions in the standard behavior. If after an interruption the robot resumes the conversation abruptly, if the response time is too high or if the robot does not respond to evident stimuli, the interaction will be negatively affected and the user will not consider it as something natural.

Apparently subtle problems can undermine this interaction and make it annoying or boring. To add more difficulty, details of all these elements as silences, interpersonal distances, the gaze direction or the forms of address can have very different meanings depending on the language, the culture, the sex or the specific person (Stivers et al. 2009).

Among all challenges with social robotics, this manuscript focuses specifically in planning the behavior of the robot at each moment. Since it is a social robot, the interaction through conversation is of capital importance. The needed deliberative process to make the robot behave coherently in one way or another depending on the situation can be solved with techniques as classical Automated Planning (Petrick and Foster 2013), avoiding to create and maintain huge finite-state machines or scripts. This manuscript also describes a real example of a social robot which uses Automated Planning and the complexity that these considerations can reach.

Related Work

Deciding the appropriate behavior of a social robot in stochastic, highly dynamic environments is a task that can be accomplished in several ways, some more convenient than others depending on the final objective. The two main aspects to consider include the way knowledge about the environment and capabilities of the robot is represented and how the reasoning using this knowledge is performed.

Knowledge Representation

Some aspects of the environment have to be represented somehow to reason about it. Humans gather information from the senses and store it in the brain in a subsymbolic way, within a neuronal network. There are robots that use this kind of knowledge representation (Baxter, de Greeff, and Belpaeme 2013; Prenzel, Feuser, and Gräser 2005). However, subsymbolic models are difficult to be reused for other solutions because they cannot be directly understandable by humans or machines, and usually they require a previous training process.

More commonly, the information to reason about the environment has been represented in a symbolic way. A direct way to work with it is through finite-state machines (Suárez-Mejías et al. 2013). In them, each state corresponds to a certain situation in which the robot can be during its execution, depending on the previous actions and the information perceived by its sensors. Each state is a combination of the modeled parameters of the environment and has a set of applicable actions. Depending on which one is executed, the robot will transit to some states or others. For simple robots this is a very fast mechanism to implement, but in more sophisticated robots it can be very hard to identify and specify correctly all possible states that could appear. Moreover, adding or modifying functionality afterwards can be very hard given that all behavior is heavily hardcoded.

When there are too many states to be maintained, there are techniques to delegate the selection of actions to an algorithm, which can be used as a black box. Automated Planning techniques are useful here. They use the description of the possible actions and the description of the initial environment to generate a plan of actions that makes the robot to accomplish some goals.

Classical Automated Planning

Automated Planning (Ghallab, Nau, and Traverso 2004), in particular action-based planning, uses two different concepts: actions and states. The execution of an action allows the transition from a certain state to another state. The objective of this technique is to find a sequence of actions to transit from an initial state to a final state in which a certain set of goals is fulfilled. A convenient way to represent this knowledge is through the domain and the problem. On one hand, the domain includes the catalog of possible action schemes, each one with preconditions that must be fulfilled in the state of the world (the modeled environment) to allow its execution and the effects in that state after its execution. On the other hand, the problem includes the description of the initial state of the world and a set of goals that must be accomplished in the final state to consider that the task is done. The domain and the problem are introduced into an automated planner that will try to find a plan of valid actions to transit from the initial state to the final state. There are many domain-independent automated planners available, most of them relying on advanced heuristic search techniques, that can be used as a black box to find a suitable plan as fast as possible. This allows the developers to focus on a higher level of abstraction; just the possible actions of the robot and the facts characterizing a state must be modeled using a symbolic language as the Planning Domain Definition Language (PDDL) (Fox and Long 2003). As a drawback, this technique is slower than domain specific techniques, like finite-state machines, so the design of the domain must be performed with care to be suitable for the fast response time required for social robots.

There are recent examples of Automated Planning for language generation and dialogue control (Steedman and Petrick 2007; Brenner and Kruijff-Korbayová 2008). Classical planning is deterministic and when interleaved with execution it requires mechanisms to recover from unexpected situations. Therefore, this work uses the planning and monitoring architecture PELEA (Alcázar et al. 2010) to con-

trol the deliberation according to the results of the execution. PELEA could be versatile enough to allow the use of learning techniques for Social Robotics (Arora et al. 2016) and plan repairing strategies (Fox et al. 2006). This kind of architecture also allows other improvements to minimize the response time of the robot by planning the first actions with precision and continue refining the final parts of the plan while the previous actions are being executed (Martínez 2016).

Much of the current research about Social Robotics relies on classical planning and replanning approaches (González, Pulido, and Fernández 2017; Chen, Yang, and Chen 2016; Vaquero et al. 2015; Rosenthal, Biswas, and Veloso 2010), as is discussed in this manuscript. There are also works about modeling social aspects of human-robot interaction for Automated Planning (Carlucci et al. 2015). However, to the best of our knowledge, none of them describes systematically the specific considerations that they had to follow while joining planning and execution to develop a competent social robot.

Every social robotic domain is hierarchical because it can be decomposed in subtasks, probabilistic because predictions of the future are needed in order to generate a plan and temporal because each action has a duration along it could be interrupted. The following subsections discuss hierarchical, probabilistic and temporal planning models and motivates the use of classical planning for Social Robotics.

Hierarchical Planning

In general terms, any complex enough activity has a hierarchic structure that is composed of a set of tasks that can be subdivided into more specific ones (Ghallab, Nau, and Traverso 2014). For instance, a social robot could have to do a questionnaire to a user. It will have to ask him several questions, each question will need to be read (through a text-to-speech mechanism) and then the answer will be heard. To read it is needed to find what to say and then play it trough speakers, etc. There are phases in the questionnaire and in the whole conversation that must be accomplished in order to finish the task correctly.

If the granularity of the deliberation is high enough then a hierarchical planner can be used directly (Nau et al. 2003). These planners use more domain knowledge to plan, sometimes improving planning times, but at the expenses of a less generic solution. Hierarchical planning also does not consider the probabilities nor the temporal aspects of Social Robotics domains.

Often, it is not needed to take deliberation to such fine-grain level and it is enough to plan at a higher level. The low level actions can be performed reactively. Depending on the robotic architecture, a component, or a reactor, could receive an action and execute it without more deliberation, like moving a robot from one point to another (Bandera et al. 2016). This has the advantage of distributing the robotic architecture into several specialized components (or reactors) for some complex reactive tasks, without having to deliberate with too much specific knowledge. This manuscript also describes a real example which uses reactors to delegate low-level actions.

Probabilistic Planning

Other planning techniques take directly into account the stochastic nature of the dynamic environments (Little and Thiébaux 2007) present in Social Robotics. A probabilistic contingent planner can plan a set of different contingent plans to be used in case that the standard plan cannot be executed due to more or less probable unforeseen events. The first drawback of these techniques in comparison to classical planning is that they can be much slower. Since the reaction time of these robots must be very fast, classical planning reaches faster planning times at the expense of a higher number of replannings. Moreover, the biggest advances in heuristics for planning are within the classical planning area, so probabilistic techniques cannot take full advantage of them.

In Social Robotics its impossible to know the exact probabilities of each effect of the action. Learning them could help to generate better plans, but at the end each person reacts in a particular way, and unpredictable interruptions can appear in any moment (as the the user leaving the room) that must be considered.

Thanks to the use of planning architectures as PELEA, it is possible to use deterministic techniques as classical planning into stochastic environments.

Temporal Planning

Temporal planning (Fox and Long 2003) uses durative actions to generate plans with concurrent actions which apply effects at the beginning or at the end of the action. Although in a social domain, as in a conversation, it is impossible to determine the duration of an action such as speak and the interaction is made in sequential steps which normally do not overlap among them, there are interesting features that can be taken from temporal planning. In particular, durative actions consider three types of conditions: “at start”, “over all” and “at end”. These conditions must be held along different moments of the execution of the action. These reasoning is needed to interrupt an action in the middle of its execution or at the end. It can be taken into account in PELEA by using specific labels directly in the PDDL code, avoiding incompatibilities between durative actions and many automated planners.

In essence, classical planning can be used along with a planning architecture as PELEA to control the hierarchical, probabilistic and temporal nature of these domains and also taking advantage of the ease of modeling of PDDL, advanced heuristics and high planner compatibility.

The Clarc Use Case

For the rest of this manuscript, the Clarc social robot (Bandera et al. 2016) is used as an example to illustrate the contributions. This section explains one of the use cases of this robot. It uses classical planning and the monitoring architecture PELEA to deliberate about its behavior.

The Clarc robot is part of an European ECHORD++ research project[1]. Its main goal is to save clinicians’ time

[1] http://www.clarc-echord.eu

by assisting elder people while performing Comprehensive Geriatric Assessment (CGA) tests to measure their general health, habits of their daily life and the ability to perform some activities without help. Many of these are questionnaire-based tests which usually are held in hospit with the assistance of a clinician. There are many differ tests to evaluate different aspects of the patients, but to b ter explain the social task, this section is focused on o two CGA tests.

Clarc 1 has a touchscreen, speakers, a microphone an 3D sensor. CGA tests are very long and heavily based speech, so the conversational requirements of this robo platform are very demanding. The text-to-speech and spe recognition mechanisms are implemented in internal co ponents of the robotic architecture, so they are out of t scope of this manuscript. Their capabilities are enough reproduce the text of each question and to recognize t needed answers. The physical embodiment of the robot c also be used to improve the interaction, but the current pro type is specially focused on the development of the conv sation for the tests. All the robot behavior has to be model within a PDDL domain.

Figure 1: The current Clarc robot prototype with a patient.

When the patient is sat in front of the robot, the clinician selects a test and the robot starts working autonomously. An easy test to introduce the basics of the use case is the Barthel (Mahoney and Barthel 1965) one. It measures the patient's autonomy in his daily life. The flow is very straightforward, every question has fixed answers that must be answered to finish correctly and return a final score. Figure 2 shows an extract of a possible initial plan of a Barthel test. As it can be seen, after configuring some elements of the test that is going to be executed, the robot introduces itself. The first parameter is a label which identifies the speech to be played through the speakers. All parameters starting with "p_" indicate the amount of time to wait until starting the next action. After introducing the test with some basic instructions, it starts reading the statement of each question slowly. The parameter "first" indicates that it is the first time that this question is read. Then it waits 10 seconds at most (as indicated by the last parameter "dur_10s") to receive the patient's answer. The executed flow of actions become more complex when the patient does not answer some questions or the robot cannot understand the answers. After a number of failed trials, the robot has to repeat the question in an al-

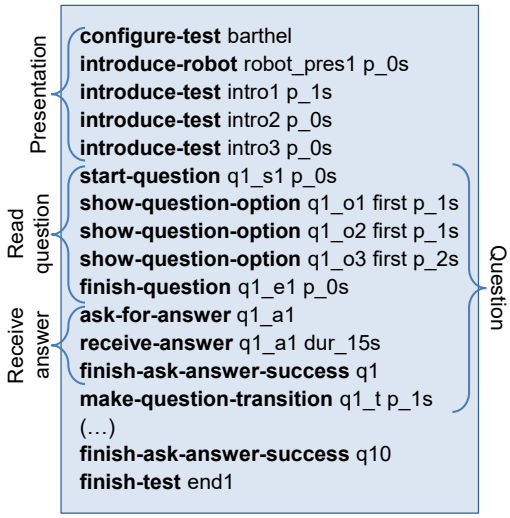

Figure 2: Example plan of a Barthel test.

All this process is executed assuming that the environment is suitable for the test. For instance, that the patient is in front of the robot, the test is not paused, there is enough battery, etc. If any of these variables change, then the execution may be interrupted and the robot must perform some corrective behavior accordingly to the detected situation. The structure of the conversation is controlled with the preconditions of the actions. For instance: after resuming the test, the robot must continue from a point which guarantees a coherent interaction with the patient, repeating some previously executed actions if needed. The timing of the conversation is controlled with the pauses and the maximum duration indicated in the parameters.

The Mini-mental test (Folstein, Folstein, and McHugh 1975) is much more complex than the Barthel one because all questions have open answers that the robot has to detect. The description provided here is to illustrate the extent of the complexity of the required behavior. The most basic questions are about the current day, month and season or the current building, city and country. The robot can detect a large set of possible answers that can be correct or incorrect. Sometimes an incorrect answer can be refined with another question for clarification. The test continues with questions about numeric operations (taking into account possible errors carried from earlier operations), repeating a list of words (altering the order is valid although not perfect), writing a syntactically correct sentence, following written commands as touching the nose and the right ear or closing the eyes, saying tongue twisters and even drawing two intersecting pentagons in the touchscreen. All this is in one long test, considering also all the corrective behaviors to manage

unexpected events.

As can be seen, each question has different interactive procedures that must be modeled in the domain that can be ruined by an unexpected interruption if it is not designed carefully. Apart from the challenges in speech recognition, computer vision and hardware coordination, the required behavior is too complex to model using finite-sate machines, so Automated Planning is useful here. However, joining the planning and the execution of the obtained plan together can be challenging too. In Clarc, the approach was to create a generic PDDL domain for questionnaire-based tests, generalizing each question as much as possible, and adapting the existing monitoring architecture PELEA to fit all communication requirements with the rest of the robotic architecture.

To design this kind of tasks properly, it is necessary to know the different issues that every developer of social robots with classical planning will need to face sooner or later. This manuscript focuses on describing these issues and their solutions.

Modeling

There are several challenges identified while modeling classical planning domains for Social Robotics. This section explains these points highlighting their impact on the interaction.

Island Domains

The social behavior has several rules that must be followed to achieve a coherent interaction. This usually implies plans that contains several ordered phases. In Clarc, for instance, these actions are the salutation, then the introductions, then the questions of the selected test and finally the farewell. We refer to these kind of domains as "island domains" where each phase constitutes a different isle.

From the planning perspective, every one of these phases can be modeled by introducing intermediate subgoals or landmarks and a mechanism to impose a total order among them. Previous work on landmarks (Hoffmann, Porteous, and Sebastia 2004) assumes that neither the landmarks nor their order are known a priori. They should be determined while planning. In "island domains" the challenge is how to build good and efficient models given that both the landmarks (phases) and their order are known.

Island domains would be very fast to plan because they are very sequential. This is important when planning for social robots, in which high planning time could undermine the interaction. However, depending on the size of problems and on how they have been modeled, planners can spend too much time on instantiation and preprocessing.

Nominal Behavior Prediction

While modeling a PDDL domain (Ghallab, Nau, and Traverso 2014) it is necessary to make some assumptions or predictions about the effects of the actions. In a real environment, these effects are stochastic because the actions can fail or an external event can change the environment. To deal with this uncertainty when planning, a "nominal behavior" can be assumed in the effects of each action, as shown in Figure 3. In the case of Clarc, this is the behavior which produces shorter plans because it is assumed that the patient will answer correctly to every question at the first try.

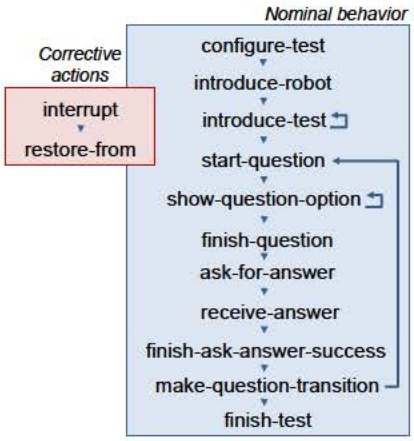

Figure 3: Possible plan of actions with the nominal behavior to perform a Barthel test in Clarc.

While executing this plan, it is very probable that the patient will fail at some point to answer a question. However, it is impossible to decide the amount of fails nor the precise questions in which they will occur. The design of this nominal behavior is at the developers' criteria. They could, for instance, to model a nominal behavior in which the patient fails each and every question, but this is less probable than answering correctly to all of them. As this is a design decision, developers could consider statistics, user preferences, etc. to increase the quality of the prediction.

Some kind of predicted behavior is needed in order to generate a valid plan. The nominal behavior can be seen as a guideline of the needed actions that are left to be executed to reach the goals.

Obviously, if the previous plan is no longer valid, another one must be generated, so replanning is considered part of the whole deliberative process even while modeling the domain.

State Decomposition

The execution of a plan requires a certain control of the external environment to check if the actions are correctly executed or not. When modeling the interaction it is important to represent both the robot internal state and the state of the environment. In Clarc, this state includes also the results of the interaction with the patient. Changes in the current state can be due to actions of the robot or to what is usually known as external or exogenous events. These external predicates can sometimes be predicted, but not always. We distinguish three different types of predicates in the state:

Internal predicates: These are used to control the domain and to organize the plan of actions. They can appear in the effects or preconditions of any action and will always have the expected value because they do not depend on anything external depending on the course of the execution. Internal predicates can represent a piece of information

as the predicate `(testIntroduction ?speech)` that indicates one of the speeches that has to be played during the introduction of a test, or they can be control predicates that denote the points of the interaction already planned (flags). Internal predicates are never changed externally during the execution.

Predicted predicates: Their value must be predicted in the nominal behavior in order to generate a complete plan, but the actual values depend only on external elements of the environment. For instance, a predicate as `validAnswer`, that represents the fact that the patient provided a valid answer to a question, needs to be predicted in the effects of an action like `CheckAnswer` to continue with the plan, following the nominal behavior. These predicates can be part of the effects and preconditions of any action, although the predicted values in the effects may differ from the actual ones obtained through the sensors.

Unpredictable predicates: These will never be part of the effects of the actions involved on the nominal behavior, although they will in the preconditions. Predicates as `pauseActivated`, representing the pause button was activated, are completely unpredictable because there is not any explicit action in the nominal behavior to change their value. A change in the value of these predicates is only triggered externally and in any moment of the execution.

This differentiation creates two types of world states, one in which the values of its predicates will be always as expected and another in which their predicates can change in any moment, invalidating the current plan in the middle of its execution.

Corrective Actions

When the expected state of the world differs from the actual state, a replanning may be needed. The new plan must contain some actions to correct the unexpected issue and to return to the normal flow of the nominal behavior. These are corrective actions which are never included in the initial plan for the nominal behavior.

For instance, Clarc must interrupt the execution of the plan if the patient leaves the test area. The new replanned plan should start calling the patient and searching for him before continuing with the rest of the test. After the execution of these corrective subplans, the nominal behavior can continue.

Modeling these corrective actions is important because it endows the system with much more responsiveness to the environment and a more coherent interaction. There is no need to increase the number of preconditions in the PDDL actions of the nominal behavior to check every considered issue. Only one unpredictable predicate in the preconditions indicating if the situation requires a corrective subplan is enough for most interactive applications. Then, the specific corrective subplan will be planned depending on the preconditions of the corrective actions.

Replanning in Interactive Steps

In any fluid social interaction there are several steps that must be followed in order. This is ensured by the nominal behavior, but when there are interruptions, the interaction can be compromised. For instance, in the Barthel test, Clarc must enumerate the options of a question to the patient immediately after reading the statement.

Suppose that the robot finishes reading the second option of a question and then, suddenly, there is an interruption that requires a replanning. The nominal behavior flow is deviated to fix the situation with a corrective subplan. After that, the execution returns to the nominal behavior, but in which point? Should the robot repeat the second option, none at all, start again from the first option or directly from the statement? Depending on the granularity of the actions, these decisions must be made thinking only in terms of interaction. Modeling them in the domain is somewhat special because the corrective plan must reset the values of internal predicates to repeat one or more previously executed actions to achieve a coherent interaction.

The number of actions to execute again after a replanning depends only in the moment in which the interruption occurs, so the execution is divided in interactive steps of different number of actions. An interactive step must be completely finished or then it has to be repeated again from its beginning. An example of these interactive steps can be seen in the sets of actions of Figure 2 like presentation, read question and receive answer.

Numerical Information

Socially interactive applications must be rich enough to be believable by their users. For instance, repeating too many times the same sentence can expose design problems in the social robot. Randomizing sentences can be easily done in a low level, but when the repetition involves a more complex behavior it should be taken into account in the planning domain.

Repetitions are only an example to illustrate the need of counting in domains for social interaction and in many other real world applications. In Clarc, especially for the Minimental test, the domain contains many numeric fluents to represent the order of the current question, the number of attempts for a question, the consecutive and total number of failed questions and so on. The use of numeric preconditions also saves much preprocessing time and simplifies the code of the domain. Modeling numbers or order relations with predicates would increase preprocessing time in Clarc to unbearable values, given that its current planning and replanning time is just below the limit for a fluid reaction in a social robot.

Using numeric fluents in big, real world domains, is easier to model, with less parameters in the actions (which implies less preprocessing time and less memory used) and there is no need to know the range of levels. Their downside is that many planners are not yet compatible with them nor have too many heuristics to ease the planning task.

For the modeling part, it is also important to note that the value of a numeric fluent could not appear in the standard output of the automated planner because it is not part of the parameters of the actions. This can be relevant while executing the plan and it is discussed later in this manuscript.

Execution

There are several specific aspects that must be taken into count to execute the generated plans into a real and dyna environment. Some of these aspects have been alre pointed out in the literature (Ghallab, Nau, and Trav 2014), but others are more specific to Social Robotics fact, the application of Automated Planning in dynamic vironments has not been studied that much, so the me nism to join planning and execution is up to each develo This section describes the main points that must be ta into account while executing the generated plans of act in a social robot.

Continuous Monitoring

The first need that arises when executing plans in dyna environments like a social robot is the ability to replan w something in the actual state of the world invalidates current plan. This can happen, for instance, when the actual state differs from the expected state during or after the execution of each action. In fact, the model of the nominal behavior and corrective actions must take replannings into account, so monitoring is of capital importance.

Monitoring the execution continuously ensures that the robot can react when something unexpected happens. For this purpose, a mature planning framework as PELEA (Planning, Execution and Learning Architecture) (Alcázar et al. 2010) can be used. Figure 4 shows the main modules of this architecture. In essence, the working flow of PELEA is as follows. The Executive module has the domain and the problem with the initial internal state of the robot in PDDL. Then, it completes the predicted and unpredictable predicates of the state with the actual ones provided by the low-level sensors of the robot (steps 1, 2 and 3). This complete high-level state is sent to Monitoring (4) to check if it is compatible with the expected state of the world. If it is the first plan or if the previous plan is not valid anymore, Monitoring retrieves a plan from Decision Support which runs a certain automated planner (5, 6). This plan is stored in Monitoring, which returns the next action to Executive (7). If, in contrast, the actual state of the world is compatible with the expected one, then Monitoring just returns the next action of the previously planned plan (skips steps 5 and 6). Finally, Executive transform this high-level action into a set of low-level actions (8, 9) and sends it to the robot (10). Then it waits until the execution is finished. After that, it completes the expected state with the information of the sensors and the cycle starts again. In this scheme, the Executive has full control of the execution, timing the maximum duration of an action, pauses, etc. to control the pace of the interaction.

The compatibility criteria of the actual and expected states of the world depend on the developer or the application. Always replanning after a single change of a predicate could lead to unnecessary replannings due to this change could not affect to any precondition of the following actions. A more relaxed comparison could be much more interesting in cases when there is much information in the state and only a part of it is relevant to interrupt the nominal behavior. For instance, the criterion in Clarc is to check if the actual state of

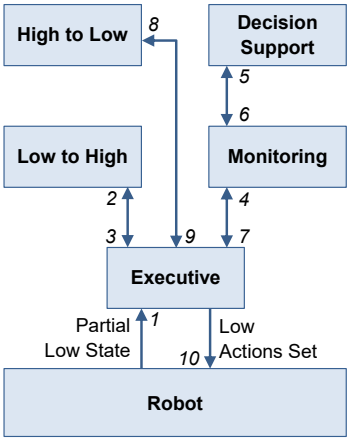

Figure 4: Basic planning and monitoring scheme of the Clarc robot with the PELEA architecture. Numbers indicate the communication flow between each module.

the world fulfills the preconditions of the next action. This criterion could not be suitable for other applications.

Also, as a side note, in these domains the goals should not be unexpectedly accomplished before finishing the execution of the nominal behavior.

Interrupting Actions

A true interactive system must be responsive in any moment of its execution. This is especially important while it is executing an action but it has not finished yet. A system that simply waits for the action to finish and then checks the state of the world is not realistic. For instance, if Clarc is executing a `Say` action, which involves a 20 seconds speech and the patient leaves the room in the 5th second, the robot must be able to interrupt the speech in the middle of its execution. If not, Clarc would continue speaking to nobody during the next 15 seconds before realizing that it has to call the patient to return to the test area.

To do this, the robot sends asynchronously the actual state of the world to Executive and then asks to Monitoring if the actual state is compatible with the action that is currently being executed. It is important to remark that the compatibility criteria after or during the execution of an action can be different. Clarc interrupts the execution if the new state violates any precondition of the action that is currently being executed. Tagging the parameters of each action in the PDDL code could easily indicate to PELEA if the value of a parameter must be the expected one only at the start of an action or during its whole execution. This is similar as temporal planning with its durative actions, but in Clarc these conditions are directly managed by PELEA because it is not needed to plan with overlapping actions.

All this indicates that it is important to pay attention to the granularity of the actions. Plans have a hierarchical nature, in which an action is later executed and refined by a component or "reactor" of the robotic architecture. More granularity of actions implies shorter ones and more autonomy for these reactors, and vice versa. Moreover, with shorter ac-

tions the domain will be more complicated in order to maintain the coherence of the interactive steps between replannings.

After the execution of an action, the robot must communicate to Executive that it has finished and that it is idle now. Then, Executive retrieves the last received actual state of the world and gives it to Monitoring to obtain the next previously planned or newly replanned action.

Planning Time Restrictions

Stochastic environments cause unpredictable situations that must be managed in these interactive systems. This means that replannings will appear, so it is very important to keep replanning times as low as possible to achieve a fluid interaction. After an interruption, the robot is going to stop in the middle of the interaction while it is replanning because it does not know the next action until it has the new plan. This means that in Social Robotics, planning must work in a way that there are not any detectable delay in the interaction.

For Clarc, a replanning time of 3 seconds is the maximum acceptable amount to achieve a fluid social interaction. Interestingly, almost all of its planning time is consumed on instantiation and preprocessing due to the large amount of different objects in the PDDL states of the world. Numeric fluents are useful to reduce the total planning time by decreasing the branching factor because they do not appear in the parameters of the actions.

Clarc, instead of stopping immediately after the triggering of an interruption, waits until the replanning is done. The previous action is interrupted only when Monitoring already has the new plan with the next action. This method avoids idle moments of the robot, allowing longer replanning times, but at the cost of slower reactions. The idea behind this is to keep the user engaged while planning proceeds.

Abstraction Levels in States and Actions

As it can be seen in Figure 4, actions and states are the main elements of information in a monitoring system as PELEA. But there are important differences in the abstraction level of these elements because the robot only works at low level and the planning system at high level. A conversion is needed to translate these actions to the robot and states from the robot. Figure 4 also shows the HighToLow and LowToHigh modules connected to Execution to manage this. This subsection expands the previous explanations detailing these abstractions:

- **High-level states**: PELEA uses them to plan. The actual state of the world is maintained by Execution and the expected state is maintained by Monitoring.
- **Low-level state**: The robot sends only a part of its low-level state to the Executive. This state always contains the latest information retrieved by the sensors and the robot. Executive sends this partial low-level state to the LowToHigh module, which abstracts this information into high-level predicates to complete the actual high-level state.
- **High-level actions**: These are the actions returned directly by the automated planner of Decision Support. Monitoring sends these high-level actions, one by one, to Executive.
- **Low-level actions**: When Executive receives a high-level action, it sends it to HighToLow to obtain a decomposition into a set of low-level actions. Then, Executive sends this set to the robot to execute these low-level actions. When all these actions are finished, the robot will communicate to Executive that it has finished the execution of the last low-level action set. Clarc considers that every low-level action in the set has to be executed in parallel, instead of sequentially. Each high-level action is modeled taking this into account because all of them have to be decomposed into a parallel set of low-level actions.

The HighToLow and the LowToHigh modules are ad hoc programmed for each domain because PDDL code cannot handle these decompositions and transformations between abstraction levels.

Retrieving the Value of Numeric Fluents

There are also some issues with the standard output of the automated planners. Numeric fluents are useful to model order relations. They do not appear in the parameters of the PDDL actions, reducing the branching factor, but this causes that their value is never shown through the output of the planner.

In Clarc, numeric values like the question number or the duration are needed when executing the actions and they are only in the expected state of the world, as the internal predicates. It uses the automated planners as black boxes, as PELEA does, so its solution to retrieve these values is generic, but inefficient. Its PDDL domain uses functions to relate the discrete values of the needed numeric fluents to predicates, inserting them into parameters in the actions to make them appear in the output of the planner. This cancels the benefits of using numeric fluents in such particular cases.

There are better solutions than increasing the number of parameters in the actions. Tagging the effects of actions in the PDDL code could be useful in this case too, and somewhat generic. It could be used to indicate that the value of certain numeric fluents must be printed after the action directly in the output plan.

Another solution could be, given the expected state, the domain and the plan, to recalculate the value of the needed numeric fluents in polynomial time. This could be done easily by parsing the output of the VAL application (Howey and Long 2003), which can be used to check the actual validity of the generated plans.

Main Performance Results with Clarc

The first prototype of Clarc included all previously discussed design considerations. The first evaluations were carried out with 24 senior end users. The main result is that the Clarc prototype finished 3 standard tests correctly, including the Barthel and Mini-mental, and reacted very fast to several unexpected events like calling the patient when he leaves the test area, offering clarifications to the patient when he suddenly asks for help or asking the patient to wait when the battery is discharged to a critical level.

Planning times increase as the length of the plan becomes longer. However, they are much lower than preprocessing times, which always remain the same for any length of the resulting plan. Figure 5 shows an example of the average times that Pelea needs to send a new planned or replanned action to Clarc. The average planning time is never higher than 3 seconds, which makes it suitable for the fast deliberations needed in this domain of Social Robotics. In the evaluation, all senior users interacted smoothly with Clarc and declared that they did not felt apprehension or discomfort with the robot.

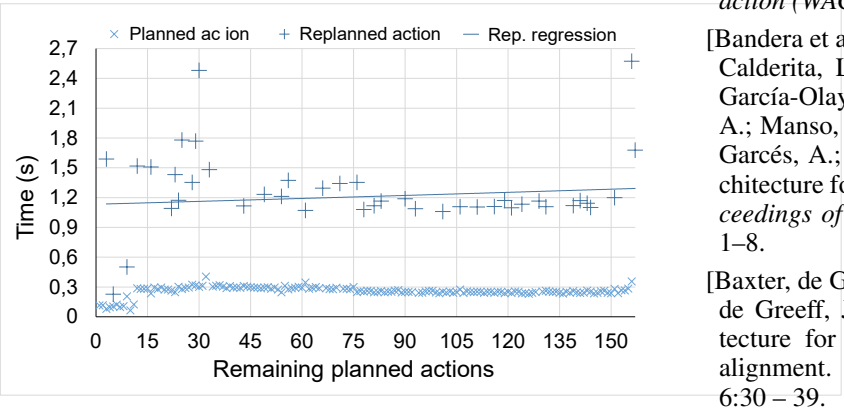

Figure 5: Average time to send a new planned or replanned action to the Clarc robot.

The next steps in the Clarc project include adding more than 10 standard geriatric tests and also customized ones by the clinician. This will require to modify and expand the Clarc domain taking the considerations described in this manuscript and check how generalizable is this solution.

Conclusions

One of the reasons behind the low impact of classical planning in real applications is the need of more research joining planning and execution. This manuscript contributes with the list of important aspects that must be taken into account while modeling a classical and deterministic PDDL domain for Social Robotics and executing the resulting plan in their highly dynamic and stochastic environments.

Constant monitoring of the execution and action interruptions are needed to be able to react to unexpected events, but fast replannings times are needed in order to achieve a fluid interaction. Numeric fluents are very important in many real problems with many objects to represent order relations without unneeded increments of the branching factor and preprocessing times.

As we have discussed here, planning, sensing and execution are interleaved tasks very dependent among them. All of them must be considered in each task when developing a complete social robotic platform.

References

[Alcázar et al. 2010] Alcázar, V.; Guzmán, C.; Prior, D.; Borrajo, D.; Castillo, L.; and Onaindia, E. 2010. PELEA: Planning, Learning and Execution Architecture. In *Proceedings of the 28th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*.

[Arora et al. 2016] Arora, A.; Fiorino, H.; Pellier, D.; and Pesty, S. 2016. A Review on Learning Planning Action Models for Socio-Communicative HRI. In *Proceedings of the 7th Workshop on Artificial Companion, Affect and Interaction (WACAI)*.

[Bandera et al. 2016] Bandera, A.; Bandera, J.; Bustos, P.; Calderita, L.; Dueñas, A.; Fernández, F.; Fuentetaja, R.; García-Olaya, A.; García-Polo, F.; González, J.; Iglesias, A.; Manso, L.; Marfil, R.; Pulido, J.; Reuther, C.; Romero-Garcés, A.; and Suárez, C. 2016. CLARC: a Robotic Architecture for Comprehensive Geriatric Assessment. In *Proceedings of the 17th Workshop of Physical Agents (WAF)*, 1–8.

[Baxter, de Greeff, and Belpaeme 2013] Baxter, P. E.; de Greeff, J.; and Belpaeme, T. 2013. Cognitive architecture for humanrobot interaction: Towards behavioural alignment. *Biologically Inspired Cognitive Architectures* 6:30 – 39.

[Brenner and Kruijff-Korbayová 2008] Brenner, M., and Kruijff-Korbayová, I. 2008. A continual multiagent planning approach to situated dialogue. In *In Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue (LONDIAL)*, 61–68.

[Carlucci et al. 2015] Carlucci, F. M.; Nardi, L.; Iocchi, L.; and Nardi, D. 2015. Explicit Representation of Social Norms for Social Robots. In *Proceedings of the 28th International Conference on Intelligent Robots and Systems (IROS)*, 4191–4196.

[Chen, Yang, and Chen 2016] Chen, K.; Yang, F.; and Chen, X. 2016. Planning with Task-Oriented Knowledge Acquisition for a Service Robot. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 812–818.

[Edelkamp and Hoffmann 2004] Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classic part of the 4th International Planning Competition. Technical report, Institut für Informatik, Freiburg, Germany.

[Folstein, Folstein, and McHugh 1975] Folstein, M.; Folstein, S.; and McHugh, P. 1975. "Mini-mental state". A practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric Research* 12(3):189–198.

[Fox and Long 2003] Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research (JAIR)* 20(1):61–124.

[Fox et al. 2006] Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan Stability: Replanning versus Plan Repair. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*.

[Ghallab, Nau, and Traverso 2004] Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. Elsevier.

[Ghallab, Nau, and Traverso 2014] Ghallab, M.; Nau, D.; and Traverso, P. 2014. The Actor's View of Automated Planning and Acting: A Position Paper. *Artificial Intelligence* 208:1–17.

[González, Pulido, and Fernández 2017] 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*. Advance online publication: http://dx.doi.org/10.1016/j.cogsys.2016.09.003.

[Hoffmann, Porteous, and Sebastia 2004] Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research (JAIR)* 22:215–278.

[Howey and Long 2003] Howey, R., and Long, D. 2003. VAL's Progress: The Automatic Validation Tool for PDDL2.1 used in The International Planning Competition. In *Proceedings of the Workshop "The Competition: Impact, Organization, Evaluation, Benchmarks", ICAPS*, 28–37.

[Leite, Martinho, and Paiva 2013] Leite, I.; Martinho, C.; and Paiva, A. 2013. Social Robots for Long-Term Interaction: A Survey. *International Journal of Social Robotics* 5(2):291–308.

[Little and Thiébaux 2007] Little, I., and Thiébaux, S. 2007. Probabilistic Planning vs Replanning. In *Proceedings of the Workshop "International Planning Competition: Past, Present and Future", ICAPS*.

[Mahoney and Barthel 1965] Mahoney, F. I., and Barthel, D. W. 1965. Functional evaluation: The Barthel index. *Maryland State Medical Journal* 14:61–5.

[Martínez 2016] Martínez, M. 2016. *Automated Planning Through Abstractions in Dynamic and Stochastic Environments*. Ph.D. Dissertation, Universidad Carlos III de Madrid, Leganés, Spain.

[Nau et al. 2003] Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research (JAIR)* 20:379–404.

[Petrick and Foster 2013] Petrick, R. P. A., and Foster, M. E. 2013. Planning for Social Interaction in a Robot Bartender Domain. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*, 389–397.

[Prenzel, Feuser, and Gräser 2005] Prenzel, O.; Feuser, J.; and Gräser, A. 2005. Rehabilitation Robot in Intelligent Home Environment Software Architecture and Implementation of a Distributed System. In *Proceedings of the 9th International Conference on Rehabilitation Robotics (ICORR)*.

[Rosenthal, Biswas, and Veloso 2010] Rosenthal, S.; Biswas, J.; and Veloso, M. M. 2010. An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 915–922.

[Steedman and Petrick 2007] Steedman, M., and Petrick, R. P. A. 2007. Planning dialog actions. In *In Proceedings of the 8th Workshop on Discourse and Dialogue (SIGDIAL)*, 265–272.

[Stivers et al. 2009] Stivers, T.; Enfield, N.; Brown, P.; Englert, C.; Hayashi, M.; Heinemann, T.; Hoymann, G.; Rossano, F.; de Ruiter, J.; Yoon, K.; and Levinson, S. 2009. Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Sciences of the United States of America* 106(26):1058710592.

[Suárez-Mejías et al. 2013] Suárez-Mejías, C.; Echevarría, C.; Núñez, P.; Manso, L.; Bustos, P.; Leal, S.; and Parra, C. 2013. Ursus: A Robotic Assistant for Training of Children with Motor Impairments. In *Converging Clinical and Engineering Research on Neurorehabilitation*, volume 1 of *Biosystems & Biorobotics*. Springer Berlin Heidelberg. 249–253.

[Tapus, Matarić, and Scasselati 2007] Tapus, A.; Matarić, M. J.; and Scasselati, B. 2007. Socially assistive robotics [Grand Challenges of Robotics]. *IEEE Robotics & Automation Magazine* 14(1):35–42.

[Vaquero et al. 2015] Vaquero, T. S.; Mohamed, S. C.; Nejat, G.; and Beck, J. C. 2015. The Implementation of a Planning and Scheduling Architecture for Multiple Robots Assisting Multiple Users in a Retirement Home Setting. In *Proceedings of the Workshop on "Artificial Intelligence Applied to Assistive Technologies and Smart Environments", AAAI*.

[Warren 2006] Warren, M. 2006. *Features of Naturalness in Conversation*, volume 152 of *Pragmatics & Beyond New Series*. John Benjamins.