Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)

Short-Term Human-Robot Interaction through Conditional Planning and Execution

Valerio Sanelli,[1] **Michael Cashmore,**[2] **Daniele Magazzeni,**[2] **Luca Iocchi**[1]
[1] Department of Computer, Control, and Management Engineering
Sapienza University of Rome, Italy
[2] Department of Informatics, King's College London, UK

Abstract

The deployment of robots in public environments is gaining more and more attention and interest both for the research opportunities and for the possibility of developing commercial applications over it. In these scenarios, proper definitions and implementations of human-robot interactions are crucial and the specific characteristics of the environment (in particular, the presence of untrained users) makes the task of defining and implementing effective interactions particularly challenging.

In this paper, we describe a method and a fully implemented robotic system using conditional planning for generating and executing short-term interactions by a robot deployed in a public environment. To this end, the proposed method integrates and extends two components already successfully used for planning in robotics: ROSPlan and Petri Net Plans.

The contributions of this paper are the problem definition of generating short-term interactions as a conditional planning problem and the description of a solution fully implemented on a real robot. The proposed method is based on the integration between a contingent planner in ROSPlan and the Petri Net Plans execution framework, and it has been tested in different scenarios where the robot interacted with hundreds of untrained users.

1 Introduction

Short-term human-robot interactions (HRIs) are useful in many application scenarios where a robot is placed in a public space interacting with non-expert untrained users.

Short-term HRI is characterized by the following aspects: (i) users are not trained and are not aware about the capabilities of the robot, (ii) each interaction is assumed to be performed with a different non-experienced user (i.e., s/he did not participate in previous interactions of the same kind), (iii) interactions are short in time. Under these conditions, the robot does not need to memorize past interactions or to adapt the current interaction based on those in the past. Moreover, user profiling is also not needed.

Although the interactions are short, it is typically necessary or convenient to generate many of them in order to realize an effective application. Thus, the problem of generating many interactions is important and the use of automated planning procedures, instead of manual definition, brings several advantages in correctness of the solution, compactness of the representation, less effort for the designer of the system, and in the general success of the overall application.

Since the interactions with non-expert untrained users cannot be predicted a priori and evaluating the complete current state is not easy for the robot, classical planning is not adequate for this process. Replanning is not sufficient for this case, as without offline reasoning about different contingencies, it is possible for the new plan to result in a dead-end state or repeated interaction (Little and Thiebaux 2007).

In contrast, conditional planning has the advantage that it does not require knowledge about a complete initial state and on-line knowledge acquisition can be performed (Hoffmann and Brafman 2005a) without requiring the computation of a complete state (Bonet and Geffner 2000).

Consequently, conditional planning is a useful technique for automatic generation of multiple short-term interactions performed by robots deployed in public spaces.

In this paper, we present a complete system where conditional planning and human-robot interaction techniques are integrated in a fully working robotic platform that has been deployed in many public environments.

The contributions of this paper can be summarized as follows: 1) a definition of the problem of generating short-term interactions as a conditional planning problem, 2) a solution of the conditional planning problem though the integration with the conditional planner Contingent-FF (Hoffmann and Brafman 2005b) in ROSPlan (Cashmore et al. 2015), 3) the integration between ROSPlan and the Petri Net Plans (PNP) execution framework, 4) implementation and testing on a real robot in several different scenarios and open-source release of the code.

The paper is structured as follows: we start with a discussion of related work in Section 2, followed by an overview of the system, in which the contributions are described. We then present the integration with the real robot and experiments in Section 4. Section 5 concludes the paper.

2 Related Work

Planning and Robotics is an area of growing interest, in which various planning techniques are being applied to new robotic domains. In particular, planning techniques

have beeen used in the control of Autonomous Underwater Vehicles and Unmanned Aerial Vehicles; and also in the domains of disaster management and manufacturing (Alterovitz, Koenig, and Likhachev 2016; Karpas et al. 2015; Mudrová, Lacerda, and Hawes 2016; Wilson et al. 2016).

The introduction of tools to facilitate the integration of planning and robotics is an important step toward furthering the use of AI systems. In this paper we use ROSPlan to integrate the use of conditional planning with our robotic platform. ROSPlan has been used in various robotic domains for this purpose (Cashmore et al. 2014; Palomeras et al. 2016; Brafman, Bar-Sinai, and Ashkenazi 2016; Cashmore et al. 2017). Moreover, the framework has been used to integrate a planner into robotic domains in which humans are present. However, planning techniques were not specifically used for human-robot interaction. (Bajones 2016)

Planning techniques have also been used in robotic applications characterized by some form of human-robot interaction.

A first group of works focuses on interactions between humans and robots and on an explicit representation of human activities in the planning formalism. For example, Answer Set programming (ASP) has been used to model an efficient planner for collaborative house keeping robots (Erdem, Aker, and Patoglu 2012). This system automatically extracts common sense information from the ConceptNet dataset and exploits it to better perform the given task. Social norms have also been considered in the planning process. For example, Human Aware Planning (Tomic, Pecora, and Saffiotti 2014) is a system based on Constraint Based Planning (CBP) that allows modeling and planning with social norms, represented as constraints of the problem. Social norms integrated into a planning system are also discussed in (Nardi and Iocchi 2014), where the output of a planner is modified in order to take into account such norms expressed as rules. Although in these works human actions and interactions are explicitly modeled, the planning process focuses on generating interactions as interleaved actions between human and robots and no actions to acquire information at execution time affecting the flow of execution of the plans are present.

A second group of works refers to human-robot collaboration and focuses on tasks in which human and robots perform joint actions in the environment. Planning techniques for human-robot collaboration have been implemented in the Hierarchical Agent-based Task Planner (HATP) system (Lallement, De Silva, and Alami 2014) that enables the definition of collaborative tasks. Recent developments of this system (Fiore, Clodic, and Alami 2016) also allows the user to interact with the robot to define the plan before its execution and the robot to infer the users' intentions. Another approach for generating plans for human-robot collaboration is described in (Milliez et al. 2015). In this work, users are characterized by the level of knowledge they have about the task and the robot acts according to it, by providing more or less information during plan execution. All these methods do not consider the possibility of having agreements about the collaborative actions at execution time. Therefore, once the plan is started, it is not possible to change its execution flow (unless with a replanning procedure).

In general, no conditional plans are used in the above mentioned papers. We choose to explore conditional planning in this context, as it is a natural way to model on-line acquisition of information during human-robot interactions explicitly as a planning problem.

Approaches based on probabilistic models (such as Markov Decision Processes (MDP) or Partially Observable Markov Decision Processes (POMDP)) can represent uncertainty in the domain and compute a policy that takes into account such uncertainty. However, in some cases, the execution mechanism still requires a complete evaluation of the current state and the system is not robust to errors in this computation. In order to overcome this difficulty, some recent approaches proposed the use of different levels of planning. For example, integration of classical planning with Partially Observable Markov Decision Processes (POMDP) is described in (Hanheide and others 2015). However, continuous switching planning methods can be impractical in a dynamic world with frequent human-robot interactions. In (Iocchi et al. 2016) a particular translation of the policies generated by an MDP solver have been integrated with a Petri Net Plan executor to reduce the need of evaluating a complete state.

However, the main problem with methods based on probabilistic models is that the solutions are sensitive to the many values of probabilities in the model that are often difficult and time-consuming to determine a priori or with learning approaches.

Conditional Planning techniques have been explored in HRI in the domain of a robotic bartender JAMES (Petrick and Foster 2013). In this application the HRI is modeled as a variant of the PDDL language (Mcdermott et al. 1998) including sensing actions. Plans for the interaction are generated using PKS (Planning with Knowledge and Sensing) (Petrick and Bacchus 2004). More recently, the planning approach for JAMES has been extended to focus on common interaction patterns, such as *clarification questions* (Petrick and Foster 2016).

We build on these techniques, aiming to provide models that are *domain independent*. That is, a more general model that can be used to represent any short-term HRI, independent from the context. In this way, the model can be populated with interaction content without any specific knowledge of planning.

The main contribution of this paper is the integration of such conditional plans for HRI with tools that make the plans more robust. Specifically, taking the ideas explored in JAMES and adding recovery procedures for circumstances that are not contained in the model.

3 System Overview

The main objective of this work is to provide a framework for the generation of robust conditional plans that can deal with uncertainty in the initial state. To achieve that, we propose a multi-layered architecture that includes a Planning System, a Plan Translator and a Plan Executor. The overall architecture is shown in Figure 1. The Planning System (Section 3.1) contains the formulation of the short-term HRI

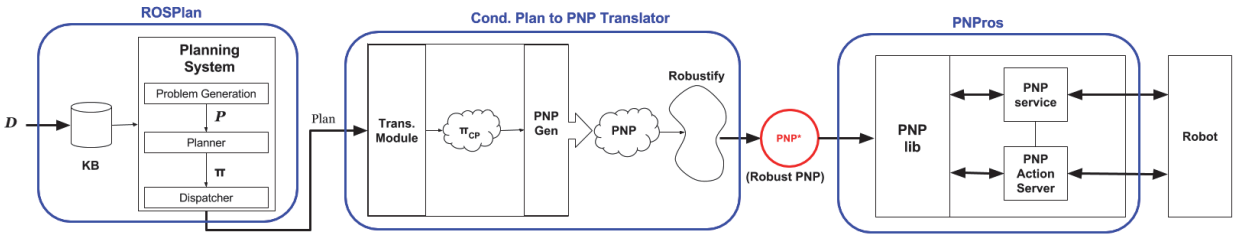

Figure 1: An overview of the system.

as a conditional planning problem and generates a conditional plan using a conditional planner. In our implementation we use the Contingent-FF planner. The plan is then translated into a Petri-Net Plan (Seciton 3.2) and finally it is executed by the PNP executor module. The following subsections describe the two novel components developed in this work: the conditional planning system and the translator to PNP. The third component is described in (Ziparo et al. 2011) and available in *http://wiki.ros.org/pnp_ros*.

3.1 Planning System

The Planning System is built upon ROSPlan (Cashmore et al. 2015). ROSPlan is a framework for embedding a PDDL planner into a ROS system. ROSPlan automatically generates a PDDL planning problem, passes this to an external planner, and dispatches the plan. ROSPlan consists of two main components, the Knowledge Base and the Planning System (see figure 2). The Knowledge Base stores the PDDL model. It stores both a domain model and the current state. It is continuously updated throughout execution, and queried by the Planning System. The Planning System integrates with the planner. We have extended the framework to generate conditional planning problems and to make use of a contingent planner. In particular, we extended ROSPlan to use the planner Contingent-FF (Hoffmann and Brafman 2005b), although other planners can be used. The plan dispatch module of the Planning System is also replaced, to instead pass the contingent plan entirely to the PNP translator.

We use the formalisation of contingent planning from Hoffmann and Brafman (2005b), where observations and uncertainty are added to (sequential) STRIPS with conditional effects.

A planning problem comprises of a domain D describing the actions and problem instance P describing the initial state and goal condition. This is extended with uncertainty about the initial state, non-deterministic action effects, and observations. The initial state is a belief state represented by a propositional CNF formula I. The possible initial world states are those that satisfy that formula. Observations are encoded as special observation actions. Such actions observe the value of a single proposition in the world state, thereby splitting the current belief state at hand and introducing two branches into the plan. Further details can be found in (Hoffmann and Brafman 2005b).

In our approach, the domain D is used for modelling the

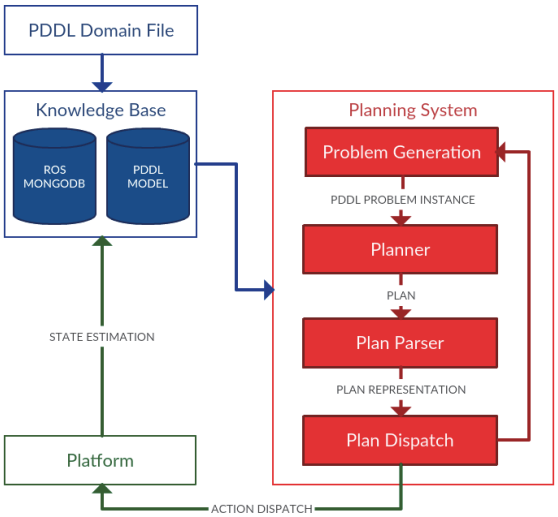

Figure 2: The Knowledge Base and Planning System components of the ROSPlan framework.

structure of any short-term interaction, independent from its specific context. The problem P characterizes the semantics of a specific interaction in a given context.

Figure 3.1 shows the conditional PDDL domain model for short-term HRI. As an example, the action `display_text` is used to display a text token to the user. This action is independent of the content of the text message. Actions `ask_question` and `check_user_response` are used to model a question and answer between the robot and the user, in which the user response is a proposition that can be observed.

Figure 4 shows a fragment of the PDDL problem for the TV Show scenario described in Section 4. The tokens (e.g., `t_news1`, `t_news2`) are entered by the domain expert, and correspond to messages specific to the context of the interaction. These are used by ROSPlan to automatically generate the PDDL problem. It is initially unknown which of the news tokens can be given to the user; this is represented using the standard syntax for uncertainty in the initial state.

Note that, while the planning process considers both the domain model and problem for generating the plan, this division is useful from the user perspective. The structural representation D requires knowledge about PDDL and planning

```
(define (domain hri_contingent)
  (:requirements :strips :typing)

  (:types
    token
    statement question - token)

  (:predicates
    (available ?t - token)
    (responded ?t - token)
    (follows ?u ?t - token)

    ;; the token might be available
    (available_to_check_s ?t - token)

    ;; the token initiates a wait
    (pauses ?t - token)

    ;; ready to give statements
    (r_HRIreceived)
    (r_HRInotreceived)
  )

  ;; Display text
  (:action display_text
    :parameters (?t - statement)
    :precondition (and
      (r_HRIreceived)
      (available ?t) )
    :effect (and
      (not (available ?t))
      (when (pauses ?t) (not (r_HRIreceived)))
      (when (pauses ?t) (r_HRInotreceived))
      (forall (?u - token)
        (when (follows ?u ?t)
        (available ?u))) )
  )

  ;; Ask for something
  (:action ask_question
    :parameters (?q - question)
    :precondition (and
      (r_HRIreceived)
      (available ?q) )
    :effect (and
      (not (available_q ?q))
      (forall (?u - token)
        (when (follows ?u ?q)
        (available_to_check ?u))) )
  )

  ;; Check user response
  (:action check_user_response
    :parameters (?t - token)
    :precondition (available_to_check ?q)
    :observe (responded ?t)
  )

  ;; Wait for screen touched
  (:action wait_for_hri
    :parameters ()
    :precondition (r_HRInotreceived)
    :effect (and
      (r_HRIreceived)
      (not (r_HRInotreceived))) )
  )
)
```

Figure 3: Conditional domain model for human robot interaction.

formalization, while P can be defined by experts of the application domain. Although not described in this paper, it is possible to allow domain experts to model the content of the interactions without any knowledge of PDDL, by providing the information in a data format that is then translated in the corresponding PDDL problem P. In other words, our formalism integrates the work of a planning expert who defines the structure of the interaction and of a domain expert who defines the content of the interactions. Various goals can be given to deliver or discover certain information, amounts of information, or observe other metrics in the interaction. The planner is then able to generate a plan according to the structure and the content specified in the model.

The planner Contingent-FF has been embedded in ROSPlan and used to generate a contingent plan for the interaction. The plan is parsed as a Directed Acyclic Graph (DAG) and published in its entirety as a single ROS message.

3.2 Plan Translator

This module parses the dispatched plan and translates it into a conditional plan. We also ensure that the obtained plan is robust, i.e. less prone to replanning/failures due to uncertainty, through a process called robustification. This is realised by specifying syntactic rules, called Execution Rules (ER), that describe possible actions and recovery procedures to perform in the plan. The overall effectiveness and robustness of the plan is increased by these routines in situations not modelled in the planning domain and possibly detected by the plan designer after its execution. The approach of improving a plan with Execution Rules has been introduced in (Iocchi et al. 2016).

In order to deal with different representations of the output plans provided by different planners, we have defined the following format to describe conditional plans:

$$\lambda = t_1; ...; t_n$$
$$t_i = a \mid < c_1?t_1 : c_2?t_2; ...; c_k?t_k >$$

In this representation λ is a plan and is composed of a sequence of terms t_i. Each term could be either an atomic action a or a multiple branch over k sensed properties $c_1, \ldots, c_k$. We assume all the conditions c_i to be mutually exclusive with each other and that each c_i is associated to a term t_i that will be executed when c_i is true. These branches are enclosed between angular brackets ($< >$) and colons (:) separate the different branches. In case conditions are not mutually exclusive and more conditions c_i are true at the time they are evaluated by the PNP engine, a non-deterministic transition towards one of the true branches will occur.

Our infrastructure includes a **Translator** module which is

```
(define (problem tvshow)
(:domain hri_contingent)
(:objects
  t_init t_name t_welcome t_intro
  t_news t_news1 t_news2 t_news3
  img_news1 img_news2 img_news3 t_joke
  t_goodbye img_coaches img_activity
    - statement
  q_activity q_news - question)

(:init
  (r_HRInotreceived)
  (available_s t_init)
  (follows_s_s t_name t_init)
  (follows_s_s t_intro t_name)
  (pauses t_intro)
  (unknown (available_s t_news))
  (unknown (available_s t_joke))
  (oneof
    (available_s t_news)
    (available_s t_joke)
  )
  (unknown (available_s t_news1))
  (unknown (available_s t_news2))
  (unknown (available_s t_news3))
  (oneof
    (available_s t_news1)
    (available_s t_news2)
    (available_s t_news3)
  )
  (pauses img_news1)
  (pauses img_news2)
  (pauses img_news3)
  ...
)

(:goal (and
  (given t_goodbye)
)))
```

Figure 4: A fragment of a problem generated for human robot interaction. The problem instance was used in the TV show scenario.

responsible for translating from standard planner output to the described representation. In particular, the planner considered in this work is Contingent-FF, as described in the previous section. This component, however, is easily extensible to consider also other planners and different output formalisms.

The algorithm that realizes the translation is described in Algorithm 1. $first()$ is a function that returns the first part of the current plan, which could be either an atomic action or a branching term. In case of atomic actions, $CondPlan\_addAction()$ creates a conditional plan state and links it to one created earlier. In case of branches, we split them and recursively construct the sub-trees. The $+$ operator represents a concatenation operator.

The module responsible for the actual generation of the PNP is called **Generator**. The algorithm that, from the conditional plan representation, generates the actual PNP is Algorithm 2. This algorithm realizes a graph visit on the structure described by the conditional plan. It keeps track of the visited states and, at each iteration, it checks the possible outcomes of the states. If the current state has no action associated with it and is not final, it keeps going with recursion. Otherwise, there are two possible cases depending on whether the current state has successors. If there are no successors, we add a final state to the PNP. If there are successors, the algorithms processes each of them. For each successor state that has not been visited yet, the algorithm adds a branch to the PNP and marks it as visited. Otherwise, if one successor state has been already visited, the algorithm adds an edge going back from that state to the current one.

Algorithm 1: Conditional Plan translation from a plan

Input : π: the plan
Output: λ: the Conditional Plan corresponding to π

Function $translCondPlan(\pi) : \lambda$
  $\sigma = first(\pi)$;
  $\pi' = rest(\pi)$;
  if $\pi = 0$ **then**
    return $\{\}$;
  else
    if $\sigma = a_i$ **then**
      $\lambda' = CondPlan\_addAction(a_i)$;
    else
      if $\sigma = \phi_1?\pi_1 : ... : \phi_n?\pi_n$ **then**
        $\Phi = CondPlan\_splitBranch(\sigma)$;
        foreach $\phi_i \in \{\phi_1, ..., \phi_n\}$ **do**
          $\lambda_i = translCondPlan(\pi_i)$;
        $\lambda' = CondPlan\_merge(\Phi, \lambda_1, .., \lambda_n)$;
  return $\lambda' + translCondPlan(\pi')$;

The complexity of the presented algorithms is linear in the number of plan states. For the translation algorithm, we generate a conditional plan state for each action in the computed plan. For the generation algorithm, we explore these states and generate the correspondent PNP. The functions used for the generation of the conditional plan and the PNP have a constant cost since they perform only local operations without looking at the whole plan.

4 Integration and Demonstrations

The presented system has been fully implemented and tested on mobile robots in several different demonstrations in public environments. The developed software[1] is based on two existing software libraries: ROSPlan[2] and PNP[3]. Also the software described in this paper has been developed under the ROS[4] middleware and it is thus suitable for integration with other robotic components.

More specifically, the developed ROS node[5] for the translation component waits for a plan on the ROS topic published by the planning system, processes this plan as described in the previous sections producing a PNP and dis-

[1] *https://sites.google.com/view/robusthri*
[2] *http://kcl-planning.github.io/ROSPlan/*
[3] *pnp.dis.uniroma1.it*
[4] *www.ros.org*
[5] *https://sites.google.com/site/rosplantopnp/*

Algorithm 2: PNPgen from Conditional Plan

Input : π: a Conditional Plan,
SS: a stack containing the places to explore
Output: π_c: PNP generated from a Conditional Plan

Function $genCondPlan(\pi) : \pi_c$
  $s = first(\pi)$;
  $\pi' = rest(\pi)$;
  $SS \leftarrow s$;
  while *(*$SS \neq 0$*)* **do**
    $s = top(SS)$;
    $a = action(s)$;
    if $(a = 0 \ \& \ s \neq s_f)$ **then**
      $genCondPlan(\pi')$;
    $O = outcomes(s)$;
    if $O = 0$ **then**
      $\pi_c = PNP\_addFinal(s)$;
      $genCondPlan(\pi')$;
    foreach *element o of* O **do**
      $s' = next(o)$;
      $\phi = obs(o)$;
      if $s' \notin SS$ **then**
        $PNP\_addBranch(\pi_c, \phi)$;
        $SS.push(s')$;
      else
        $PNP\_GoBack(\pi_c)$;

patches the PNP to the pnp_ros module for execution. This process can be executed both off-line or on-line during the execution of a robot task. In the on-line case, the pnp_ros module replaces the current plan with any new-coming plan. The planning process for the use cases discussed below is very fast: for these problems, less than a second is required. Moreover, in comparison, the translation from plan to PNP takes negligible time.

4.1 Use cases

The proposed system has been tested in different environments characterized by short-term human robot interactions with non-expert users. The plans defined and generated with the proposed system have thus been exposed to execution in a real environment. Almost all interactions have been correctly handled, regardless the lack of experience of the users interacting with the robot, thus showing the effectiveness of the proposed method.

Videos showing the interactions that occurred in these demonstrations are available at *https://sites.google.com/view/robusthri/video*. In all the examples shown, the robot executes short-term interaction plans generated with the system described in this paper. While the navigation of the robot in the environment was manually driven, mainly for safety reasons, all interactions that required input from the users (through speech or using the GUI on the tablet of the robot) have been modeled

Figure 5: A group of users interacting with the robot at Maker Faire 2016.

Figure 6: A user interacting with the robot during the Erasmus welcome day.

and implemented as sensing actions. Similarly, the flow of execution of the plan is chosen at execution time according to the outcome of such interaction (i.e., sensing actions).

1. TV Show. The first demonstration described in this paper was performed during a live TV show in which the robot was interviewed alongside a human. Before going to the TV studio, a sketch of the demo was prepared and it was presented to the TV show director one hour before the show. The director made some comments about the interactions and asked to modify something and to be flexible with respect to timing of the show. Before the live show, the plan was modified and during the show the conditional plan possessed the requested flexibility.

A Conditional Plan is generated by Contingent-FF for this problem and converted into a DOT graph, as shown in Figure 10. This plan is then translated into the intermediate format shown in Figure 7. Note that the extended PDDL used by Contingent-FF can only produce binary branches upon sensing actions, such as in `check_user_response`. These actions do not represent any action taken by the robot, but instead signify a branch point in the plan. As a result they do not appear in the translated plan.

```
display_init;
waitfor_start;
display_image_coaches;
display_text_name;
display_text_intro;
waitfor_screentouched;
display_image_activity;
ask_whichactivity;
waitfor_HRIreceived;
< news ?
        display_text_news;
        ask_whonews;
        waitfor_answer_@Q;
        display_image_answer_@Q;
        display_text_answer_@Q:
  joke ?
        display_text_joke
>;
waitfor_screentouched;
display_image_coaches;
display_text_goodbye;
restart
```

Figure 7: A conditional plan for the TV Show.

2. Maker Faire. In this demonstration, the robot was deployed during a Maker Faire for three days to interact with people; children in particular. The interaction contained different choices: information about the robot, quizzes, and jokes. Also in this example, the robustification phase added some recovery procedures to take into account some unexpected behaviors of the users.

During the three days of the event, we needed to change the content several times. For example, (i) we realized that some interactions were too long and we wanted to simplify them, (ii) we wanted to change some of the content of the interactions for the day in which schools were present, and (iii) we wanted to test different interaction modalities.

To this end, the use of a planning system was fundamental in order to generate new plans on the fly when needed, by just changing the specification of the problem.

During these demonstrations, the robot interacted with about 300 people. Most of the executions were successful, while a very few executions failed and required a manual restart of the system. These situations were due to unexpected interactions, such as, for example, users closing the window running the interaction program or restarting the operating system on the tablet.

An example of conditional plan used in this example is shown in Figure 8, in the format generated by the Plan Translator module.

3. Erasmus welcome day. Finally, the system was tested during a Faculty meeting for Erasmus students. Short-term interactions were designed to welcome the students, to ask the students from which country they come and to show some information about the countries. Since the list of students and thus of countries was known in advance, it was possible to pre-define all the possible countries and to generate appropriate questions and answers for each country.

```
display_init;
waitfor_answera;
askimg_whichinfoABCD;
waitfor_HRIreceived;
<
 answera ?
          display_image_info_alfred;
          display_text_info_alfred:
 answerb ?
          display_image_info_r20;
          display_text_info_r20:
 answerc ?
          choose_joke;
          waitfor_joke_@J;
          display_text_joke_@J:
 answerd ?
          display_image_info_roberto;
          display_text_info_roberto
>;
display_text_goodbye;
restart
```

Figure 8: A conditional plan for the Maker Faire.

```
display_init;
waitfor_personhere;
display_text_welcome;
ask_whichcontinent;
waitfor_continent_@C;
ask_whichcountry_@C;
waitfor_country_@N;
display_text_@N;
display_image_@N;
ask_wantphoto;
waitfor_no;
display_text_greet;
display_text_goodbye;
display_init
```

Figure 9: The plan for the Erasmus Welcome day.

This process was done by a semi-automatic tool for the generation of interactions and is not described in this paper.

The robustification phase in this example has been implemented with recovery procedures acting when users do not answer a question after a timeout and restarting the interactions in these cases.

About 40 students were present in the event and at least half of them interacted with the robot. Since user studies were not an experimental focus, we have not collected specific data about such interactions, but just observed that all the executions of plans were correctly completed (in some cases, thanks to the recovery procedure).

The plan used in this demonstration is shown in figure 9. This format is generated from the DAG by the Translation Module, and then encoded into a PNP, as described in Section 3.2.

5 Conclusions

In this paper, we described a method and a fully implemented robotic system using conditional planning for gen-

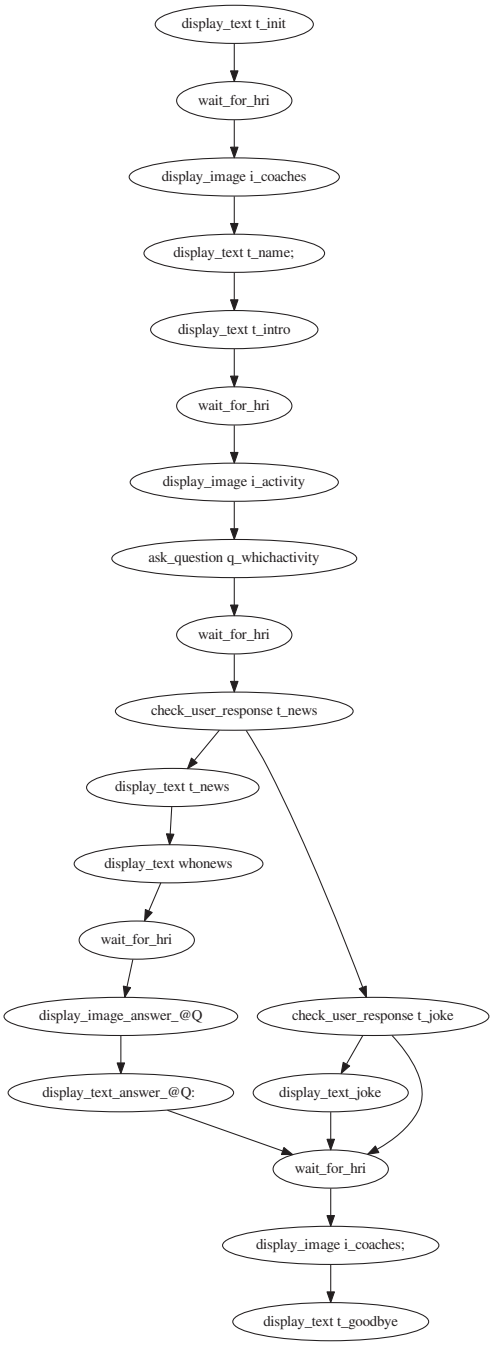

Figure 10: The conditional plan generated by Contingent-FF in DOT form.

erating and executing short-term interactions by a robot deployed in a public environment.

While conditional planning has previously been used in short-term HRI, our focus is on 1) the separation of the structure of the planning model and the context of the interaction, 2) the integration into an execution environment through the ROSPlan and Petri Net Planning frameworks. The system has been fully implemented and tested on a real robot in several different scenarios and open-source release of the code.

Our extension integrates conditional planning into ROSPlan. This will allow the generation and execution of conditional plans on board a robot, through the ROSPlan framework. Moreover, it is now possible to use a contingent planner to generate the conditional plans used as input to PNP. The result is a powerful and flexible tool for conditional planning on board robotics systems, with potential beyond short-term HRI.

In future work we intend to formalize the conditional planning domain used for our examples, providing links between the *common interaction patterns* and the general structures in our domain.

Acknowledgements

This work has been partially developed within the COACHES project. COACHES is funded within the CHIST-ERA 4^{th} Call for Research projects, 2013, Adaptive Machines in Complex Environments (AMCE) Section. Sapienza University is funded by MIUR (Italy). This work has been partially funded by European Commision as a part of SQUIRREL project that involves King's College London, under grant agreement No FP7-610532.

References

Alterovitz, R.; Koenig, S.; and Likhachev, M. 2016. Robot planning in the real world: research challenges and opportunities. *AI Magazine* 37(2):76–84.

Bajones, M. 2016. Enabling long-term human-robot interaction through adaptive behavior coordination. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*, 597–598. IEEE Press.

Bonet, B., and Geffner, H. 2000. Planning with incomplete information as heuristic search in belief space. In *Proc. of the Conference on AI Planning and Scheduling*, 52–61. AAAI Press.

Brafman, R. I.; Bar-Sinai, M.; and Ashkenazi, M. 2016. Performance level profiles: A formal language for describing the expected performance of functional modules. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2016, Daejeon, South Korea, October 9-14, 2016*, 1751–1756.

Cashmore, M.; Fox, M.; Larkworthy, T.; Long, D.; and Magazzeni, D. 2014. AUV mission control via temporal planning. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 6535–6541. IEEE.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the robot operating system. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 333–341.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; and Ridder, B. 2017. Opportunistic planning in autonomous underwater missions. *IEEE Transactions on Automation Science and Engineering*.

Erdem, E.; Aker, E.; and Patoglu, V. 2012. Answer set programming for collaborative housekeeping robotics: representation, reasoning, and execution. *Intel Serv Robotics*.

Fiore, M.; Clodic, A.; and Alami, R. 2016. On planning and task achievement modalities for human-robot collaboration. In *Experimental Robotics*, 293–306. Springer International Publishing.

Hanheide, M., et al. 2015. Robot task planning and explanation in open and uncertain worlds. *Artificial Intelligence*.

Hoffmann, J., and Brafman, R. 2005a. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 71–80.

Hoffmann, J., and Brafman, R. 2005b. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 71–80.

Iocchi, L.; Jeanpierre, L.; Lázaro, M. T.; and Mouaddib, A.-I. 2016. A practical framework for robust decision-theoretic planning and execution for service robots. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 486–494.

Karpas, E.; Levine, S. J.; Yu, P.; and Williams, B. C. 2015. Robust execution of plans for human-robot teams. In *ICAPS*, 342–346.

Lallement, R.; De Silva, L.; and Alami, R. 2014. HATP: An HTN planner for robotics. In *2nd ICAPS Workshop on Planning and Robotics*, 20–27.

Little, I., and Thiebaux, S. 2007. Probabilistic planning vs. replanning. In *In ICAPS Workshop on IPC: Past, Present and Future*.

Mcdermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. Pddl - the planning domain definition language. Technical Report TR-98-003, Yale Center for Computational Vision and Control,.

Milliez, G.; Lallement, R.; Fiore, M.; and Alami, R. 2015. Using Human Knowledge Awareness to Adapt Collaborative Plan Generation, Explanation and Monitoring. In *International Conference on Human-Robot Interaction*.

Mudrová, L.; Lacerda, B.; and Hawes, N. 2016. Partial order temporal plan merging for mobile robot tasks. In *Proc. of the 22nd European Conf. on Artificial Intelligence (ECAI)*.

Nardi, L., and Iocchi, L. 2014. Representation and execution of social plans through human-robot collaboration. In *Fifth International Conference on Social Robotics (ICSR 2014)*, 266–275.

Palomeras, N.; Carrera, A.; Hurtós, N.; Karras, G. C.; Bechlioulis, C. P.; Cashmore, M.; Magazzeni, D.; Long, D.; Fox, M.; Kyriakopoulos, K. J.; et al. 2016. Toward persistent autonomous intervention in a subsea panel. *Autonomous Robots* 40(7):1279–1306.

Petrick, R. P. A., and Bacchus, F. 2004. PKS: Knowledge-based planning with incomplete information and sensing. In *Proceedings of the System Demonstration session at ICAPS*.

Petrick, R. P. A., and Foster, M. E. 2013. Planning for social interaction in a robot bartender domain. In *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS*.

Petrick, R. P. A., and Foster, M. E. 2016. Using general-purpose planning for action selection in human-robot interaction. In *Proceedings of the AAAI Fall Symposium on Artificial Intelligence for Human-Robot Interaction (AI-HRI)*.

Tomic, S.; Pecora, F.; and Saffiotti, A. 2014. Too cool for school - adding social constraints in human aware planning. In *Proc. of 9th International Workshop on Cognitive Robotics*.

Wilson, M. A.; McMahon, J.; Wolek, A.; Aha, D. W.; and Houston, B. H. 2016. Toward goal reasoning for autonomous underwater vehicles: Responding to unexpected agents.

Ziparo, V.; Iocchi, L.; Lima, P.; Nardi, D.; and Palamara, P. 2011. Petri Net Plans - A framework for collaboration and coordination in multi-robot systems. *Autonomous Agents and Multi-Agent Systems* 23(3):344–383.