Health, demographic change and wellbeing

Personalising health and care: Advancing active and healthy ageing

H2020-PHC-19-2014

Research and Innovation Action

Deliverable 2.6

Activity plans representation

Deliverable due date: 05.2017	**Actual submission date: 31.05.2017**
Start date of project: February 1, 2015	**Duration: 42 months**
Lead beneficiary for this deliverable: UNITN	**Revision: 1.0**
Authors: Paolo Bevilacqua, Daniele Fontanelli, Marco Frego, Thomas Given-Wilson, Sean Sedwards	
Internal reviewer: Luigi Palopoli	

The research leading to these results has received funding from the European Union's H2020 Research and Innovation Programme - Societal Challenge 1 (DG CONNECT/H) under grant agreement n°643644

Dissemination Level

R	Restricted	
CO	Confidential, only for members of the consortium (including the Commission Services)	**X**

Contents

1 Executive Summary

This deliverable concerns the representation of activity plans, which must be adequately expressive to accommodate probabilistic information and verification, while being efficient to implement. In this work we thus analyse and describe the syntactic and semantic representation of activity plans. We first survey various motion planning languages, to identify the most suitable in our context. We then present case studies relevant to ACANTO and use these to motivate our choice of planning language. We then explain how the semantics of activity planning will be implemented and how it links to the syntactic representation. We conclude by extending our planning model to introduce the support for group activities.

In summary, we have identified the Planning Domain Definition Language (PDDL) language as a suitable means to represent activity plans at a syntactic level. We show that this language is adequately expressive to represent probabilistic information and other annotations necessary for the activity planning required by the ACANTO case studies. We describe how we intend to abstract the environment as a graph where the nodes represent the set of possible Points of Interest to visit. In this way, the motion actions associated to an activity plan correspond to edges connecting pairs of Points of Interest, to which we can give various weights, corresponding to different metrics related to the properties and constraints of an activity. The planning of activity thus corresponds to finding a path in the abstract graph visiting a subset of the points of interests, while fulfilling all the hard (and possibly also the soft) constraints. The metrics will also be used to monitor the progress of the user with respect to an activity and anticipate when the activity needs to be re-planned or terminated.

This document reports the final outcome of our efforts, which started in P1 with the production of a preliminary report (D2.5). The most important novel contributions contained in the present document are: 1. We make our final choice on the language used, whereas in the previous version we just studied a range of possible options, 2. We have completely defined a general domain for activity definition using the PDDL language (in the first version, we had just shown a small-scale example), 3. We have extended and generalised the model to encompass the case of groups of users, 4. We have updated the two last sections to reflect the changes made in the Activity Monitor and in the Activity Planner.

2 Introduction

The main objective of Task 2.3 is to provide an effective means to represent the activity planned for the user. The language to synthesize and represent the activities has to be flexible enough to comprise the following features: a) probabilistic planning and different levels of rewards according to the user experience and planned exercise; b) synthesis by means of control actions (which is another outcome of Task 2.3 and discussed in this deliverable); c) monitoring capability of the planned activity; d) possibility to measure the maximum deviation degree from the planned activity.

Figure 1 illustrates the basic elements and interactions of our proposed motion planning architecture. The Activity Planner constructs an Activity Plan from a palette of alternatives, according to the requirements it is provided (e.g., amount of exercise, undesirable areas). The Reactive Planner monitors the user's physical location with respect to the plan, other moving and static objects in the environment, and the constraints it is provided (e.g., desirable proximity to others). As a result, it suggests a notionally optimal direction of movement to the user. The user may not necessarily follow the suggestions, but at all times the Sensors calculate the user's actual position with respect to surrounding objects. If the user strays too far from the plan for a long time, the Motion Execution Monitor component of the Reactive Planner informs the Activity Execution Engine, that may require a re-planning of the whole Activity. The Activity Execution Monitor monitors the user's progress with respect to the high level goals of the activity (e.g., maximum elapsed time). Like the Reactive Planner, if at any time the Activity Execution Monitor judges that the goals of the Activity Plan have not or will not be achieved, it will request a re-plan from the Activity Planner.

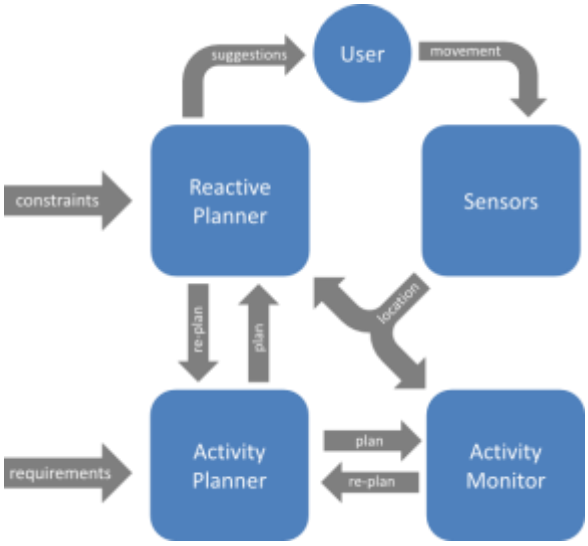

Figure 1

In what follows, we first present an overview of the possibilities available for planning in similar contexts and draw our conclusions presenting the ACANTO choice using specific case studies. We then relate the syntactic choice to the underlying semantics of motion planning and describe how activities will be executed and monitored. Finally, we summarise our approach and highlight areas of ongoing research.

3 State of the art

In the realm of planning, the problem of finding a suitable sequence of control actions that leads an electromechanical platform from an initial configuration to a desired configuration is usually referred to as *motion planning*, which involves systems with continuous dynamics given as ordinary differential/difference equations. To highlight this characteristics, motion planners are usually called continuous state space motion planners [3], less often control problem in Control Theory [4]. The distinctive feature of these approaches is that they are usually quite easy to be expressed, but very difficult to solve. This peculiarity gave birth to a plethora of solutions that treats the motion planning problem in combination with constrained motion areas, e.g., navigation functions for sphere workspace and obstacles [5], potential field-based control algorithm for workspace with geometric

partitions [6], sampling-based motion planning techniques like probabilistic roadmap method [7], and rapidly-exploring random trees [8, 9].

A radically different approach has been historically followed by the Artificial Intelligence community [10], in which the planning problem is seen as a sequence of possible *control actions* that act on the system and change its configuration. In such a case, the planner does not deal with a motion planning problem but, more appropriately, with a *task planning problem*. The greatest distinction between motion and task planning is that the state space in motion planning is continuous and possibly unbounded, making it impossible to enumerate all the states explicitly as in task planning. Usually, these problems are manageable whenever a finite, well established set of control actions is available. This makes another remarkable different with motion planners where the set of allowed actions is infinite given the continuous input space. Each control action is described by:

The *precondition* that has to be fulfilled before the action can be performed;

The *effect* on the system state after performing the action.

Again, since for motion planners the dynamical system may evolve differently under the same actuation signal from different initial states, the notion of the action conditions and effects are not well defined. For task planners, instead, the system is modelled as a discrete state transition system [11] and different states representation are available, which results in different complexities when solving the planning problem. Logic-based representation is currently one of the most popular formalisms used by many planning tools, like STRIPS [12] and PDDL [13]. An interesting feature of these approaches is that the solving process is similar to human deliberation that chooses and organizes actions by anticipating their outcomes.

STRIPS representations are based on the definition of initial and goal states, and on actions having pre-conditions and post-conditions (effects), defined using propositional logic formulae. Over the years, new formalisms and tools have been defined to model problems of increasing complexity, with the aim of solving problems closer to real world scenarios, and with practical applications. A first improvement is the Action Description Language (ADL) [47], introducing the possibility to define first-order logic predicates, and actions having conditional effects, happening only when individual preconditions for each effect hold. [48]. Further tools have been proposed to handle temporal actions and tasks, and to model numeric properties evolving over time. For example, the extension of PDDL 2.1 [13] introduces numeric fluents and durative actions to model these elements. Another solution working with temporal domains is IxTeT [49], defining a language similar to PDDL2.1, but more powerful, as it gives the possibility to access specific time points within the execution of durative actions.

Other languages have been developed to support planning problems with specific domains. For example, the NASA has developed the NDDL [50] and then the ANML (Action Notation Modeling Language) [51] as languages for planning missions and activities. Both these languages allow the expression of temporal relations and constraints. In particular, the ANML, similarly to the PDDL, is based on the concepts of actions and states, but supports valued variables and rich temporal constraints. In addition, it provides a simple way to model resources and their usage (e.g. fuel, energy).

The introduction of additional formalisms has increased the expressiveness of planning languages, giving the possibility to model soft and hard constraints on the trajectory of the plan, to model partial observability, the presence of different initial states, and of actions having different, non-deterministic effects [52, 53]. Finally, solutions have been proposed to model probabilistic problems like for example Markov Decision Processes, where the effects of actions have probabilistic distributions, and state transitions may produce different rewards. Usually, the aim of this kind of problems is to maximize the expected reward, as for example in the probabilistic extension of the PDDL [54]. Another language developed quite recently as an extension of both the PDDL and the Probabilistic-PDDL is the RDDL (Relational Dynamic Influence Diagram Language) [55]. It allows the modelling of nonlinear difference equations and unrestricted concurrency. Moreover, it supports partial observability and exogenous effects having some known probability distributions.

The most interesting feature of task planners is their ability to enumerate all the motion possibilities by means of control actions. This fact opens to the application of model *checking* techniques for the synthesis of the planning path. Given a finite transition system modelling the actual hardware or software system or its abstraction, a model checker exhaustively and automatically verifies if the given model satisfies a given specification, which normally involves security requirements such as absence of bad states and deadlocks [14, 15, 16]. The automaton-based model-checking algorithm

returns either success indicating that all possible system behaviours satisfy the property, or a counterexample as one possible behaviour that fails the property. The propositional and temporal requirements on the system behaviour are specified using *temporal logic language*, such as model-checking algorithms have been conceived for verification, they are becoming attractive also for synthesis. In fact, the purpose of verification is to find any system behaviour that violates the property, therefore for the synthesis we are interested in finding one of the system behaviours satisfying the property, and more importantly that is optimised regarding certain cost functions. There has been many recent work that integrates motion planning algorithms with model-checking techniques to treat complex motion tasks specified by temporal logics [17, 18, 19]. In such a case, temporal logics such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) have been found very efficient in providing a formal and high-level way of describing motion objectives that are much more complex than the classical point-to-point navigation problems, e.g., coverage of several regions, sequential visits, response or surveillance. The possibility of applying model checking techniques to path synthesis have been investigated in [20] during the DALi project [2].

The model-checking-based controller synthesis has been applied to both complex dynamical systems and autonomous robots: [21] considers the control strategy synthesis for discrete-time linear systems where the control objective is given by linear temporal logic formulas, beyond the usual stabilization and output regulation objectives; [22] proposes an optimal control strategy for discrete-time systems under the temporal logic constraints and a quadratic cost function; [23] and [24] present a computational framework for the a feedback control strategy synthesis of discrete-time piecewise affine systems, where the specification is given as linear temporal logic formulas over linear predicates; [25] provides a robustness index for the satisfiability of continuous-time signals under temporal logic specifications; [26] synthesizes a generic cyber-physical system with respect to a metric temporal property. Fainekos et al. [27] firstly proposes the complete framework of automated controller synthesis for autonomous robots under temporal logic tasks. The robot's motion is abstracted by its dynamical transitions within a partition of the workspace, as a finite transition system. Then a high-level discrete plan as a sequence of regions to visit is synthesized by the modified model-checking algorithms, which is then implemented by low-level continuous controller executing the control actions.

One major drawback of the proposed approaches, that are computed off-line at the beginning of the mission, is their inability of reacting in real-time to changes in the environment. Indeed, the basic assumption is that the workspace is perfectly known and is correctly represented in the finite transition system [28]. Some extensions consider the incomplete representation of the workspace [29] by modelling the robot's motion in the uncertain surrounding environment using a nondeterministic Markov decision process, where the goal is to find a control strategy that maximises the probability of satisfying the specification. Other approaches use a game-theoretic approach to model the game between the robot and the environment and then synthesise a strategy for the robot by exhaustively searching through all the possible combinations of the robot movements and the admissible workspaces [30], [31]. Instead of aiming for an off-line motion plan considering all the possibilities, [32] proposes to create a preliminary plan based on the initially available knowledge about the robot and the workspace and then use real-time observations about the workspace and the plan monitored execution to verified and, in case, revise the original plan. Similar ideas appear in [33] by locally patching the invalid transitions.

The approach taken in ACANTO in this respect is different, since the goals of the Activity Planner and Reactive Planner are significantly distinct and we do not attempt to use them both to achieve the same goal. The Activity Planner is concerned with completing high level activities under various constraints, such as distance from a bathroom, ideal calorific burn, achievement of major milestones, minimum pace. The Reactive Planner is instead focused on reaching the next waypoint in the activity, trying to avoid collisions and maintaining group cohesion. Hence, we delegate high level planning, constraints and goals to the Activity Planner, and otherwise operate in a monitoring role with the Reactive Planner, handling the finer details. For example, the Activity Planner may pick a path through the mall and inform the Reactive Planner of the next waypoint. However, significant deviation to avoid collision (or violating an Activity Planner constraint) can force re-planning. In the other direction, the motion taken by the user is reported back to the Activity Monitor to account for the actual costs/constraints rather than those projected in planner (e.g. a path may be projected to cost 110 calories, but the user expends 120 calories).

In fact, the classic motion planning framework presented previously framework reports a failure when the given task specification is not realisable [34]. To have a measure of the deviations from the original plan, [35,46] suggests to address the problem in a systematic and find the feasible relaxed specification automaton that is closest to the original one, while [36] introduces a way to determine the cause of an infeasible task analysing the environment and the system jointly. [37, 38, 39] follow a different approach by providing a way to synthesise a feasible path that "fulfills the most" the infeasible task. These approaches align with the proposed approach above for ACANTO, in order to impose further constraints to the Reactive Planner (Task 5.2) using the information of the Activity Monitor (Task 5.3).

Another important research thread for planners is related to the intrinsic distributed nature of the application. Indeed, there is an increasingly number of applications in which a set of cooperating agents coexist in the same environment to reach a common goal. Two main paradigms can be distinguished for planning algorithms. On one hand, there is a tightly-coupled and top-down approach, where the overall plan is split into a set of sub-tasks and then each agent is assigned one of the identified sub-tasks and then the agents execute them in a synchronized manner [40, 41]. On the other hand, there is a loosely-coupled and bottom-up approach in which the cooperation is intended as an emerging behaviour of the team. In practice, there is no global task and each agent acts according to its own plan in order to accomplish the task, meanwhile it interacts with other agents when necessary, in terms of information exchange and collaborations.

Our application in the ACANTO project fits into the second group, since each user must be able to plan independently (especially if connection to the infrastructure is lost), and needs to know about or consider the plans of all other users in the vicinity. This is further reinforced by recognising that while groups may be able to share some common planning information (particularly at the activity level), in general many agents will not share identical constraints, and goals; and that this information should not necessarily be disseminated.

Interestingly, the model checking-based control synthesis framework has also been extended by many research teams to a multi-agent system consisting of autonomous agents. Among the others, [42] and [43] decompose a global task specification to bi-similar local ones in a top-down manner, [44] deals with distributed optimal path synthesis under traveling time uncertainties given an assigned global goal. Linear temporal tasks have been instead considered for a multi-UAV routing problem by [9]. A totally decentralised approach is instead followed by [37] and [45], where a team of cooperative agents with different, independently-assigned, even conflicting individual tasks is considered.

Based on the analysis here presented, the following table reports a summary of the solutions that best fits the the activity planner requirements of ACANTO.

FORMALISM	*FEATURES*
STRIPS	Simple formalism. Definition of initial and goal states. Definition of propositional variables. Definition of actions, having preconditions (must hold) and postconditions (effects), involving the propositional variables.
Action Description Language (ADL)	Action language like STRIPS. Possibility to define first-order logic predicates. Possibility to define conditional effects, happening only when some preconditions hold.
IxTeT	Language based on actions (tasks), with multi-valued domain attributes. Possibility to define temporally qualified conditions and effects (events), happening at specific points during the execution of an action. Possibility to express temporal constraints regarding each task.
NDDL	Language to model hybrid systems. Based on the definition of variables and constraints. Support for user defined classes. Support for temporally qualified states and actions. Possibility to define temporal constraints and relationships between different actions.
ANML	Based on the modelling of actions and states. Support for multi-valued variables. Support for rich temporal constraints. Support for action conditions, effects and resource usage. Support for HTN (Hierarchical Task Networks) decomposition. Specific formalisms to model resources (like battery charge) and their use
PDDL 1.0	Possibility to define problems making use of different features, by indicating for each problem the needed requirements. Based on states and actions. Support for ADL conditional effects. Possibility to define a domain file used for multiple problems.
PDDL 2.1	Extension of PDDL 1.0. Support for numeric properties evolving over time, having discrete or continuous changes Support for temporal problems through durative actions Support for plan metrics, to define some numerical quantities that must be maximized/minimized
PDDL 3	Extension of PDDL 2.1 levels 1 and 2.

	Support for trajectory constraints that must be respected by the generated plan Support for preferences (soft constraints) Soft constraints can have different priorities, causing different penalties when violated
Probabilistic PDDL	Extension of PDDL 2.1. Support for probabilistic effects (different outcomes having different probabilities). Introduction of a specific fluent, reward, to model Markovian rewards, associated to each state transition. The plan must maximize the probability of reaching the goal state, or the expected value of the reward.
Multi Agent PDDL	Extension of PDDL 3. Introduces the support for plans with multiple agents. Possibility to model interactions and collaborations between the different agents.
RDDL	Extension of both PDDL and Probabilistic PDDL. Support for nonlinear difference equations and unrestricted concurrency. States, actions and observations are parametrized variables. Evolution over time modelled by stochastic functions to define the value at the next state of each variable, depending on the current values of state and action variables.

4 Case studies

To illustrate our choices more concretely in Section 5, in this section we briefly recall three use cases relevant to ACANTO, as idendified by WP1.

First Use Case

Isabel is an 82 year old woman who has lived alone for the past two years. She lives in a flat by herself in Newcastle. She no longer goes out very often and has become very physically inactive, even if her doctor suggested to stop with this unhealthy behaviour. Her daughter brings her groceries once a week. While Isabel used to enjoy going for walks, she no longer has anyone to go walking with. She recalls the times she spent walking with fondness and wishes that she had someone to go walking with. One day, she receives an invitation by mail to try out the new *FriTab* and *FriWalk* system. After receiving the system when a researcher visits her, she tells the system about her background and interests. She tells the system that she used to enjoy walking. Later that day, the *FriTab* suggests that she meet a lady in the next street, Martha, who likes to visit the local shopping mall and has the same platform. The system has noticed that Isabel does not have many friends and believes that if she had a friend who also enjoyed going for walks, she might go there again. Isabel is hesitant at first but then agrees to meet up and try out the *FriWalk*. The *FriTab* tells Isabel to meet Martha the following Wednesday at 10am to enjoy a morning together at the shopping mall. Once she arrives to the shopping mall, the *FriWalk* shows the directions to get in touch with Martha at the prescribed time. Since Martha has a similar *FriWalk*, the two ladies meet with any problem. Isabel and Martha go for a walk in the mall and decide to buy some groceries on their own. The *FriWalk* suggests the route and monitor the execution of the activity, in order to report to her medic the physical activity that has been carried out. During the walk, the *FriTab* realises that Martha feels a little bit tired and suggests to interrupt the planned activity. The *FriTab* suggests Isabel and Martha to have lunch in a local cafe. The *FriWalk* devices guides them gently to the desired cafe where they have a pleasant lunch and agree to meet up again in the following days.

Second Use Case

Michael is a 72 year old man who lives alone in Felling, Gateshead. For the past few years, he has found mobility very difficult and he is waiting for a hip operation. Consequently, he doesn't get out much. He used to enjoy visiting museums and now fulfils his passion for natural history by watching documentaries on TV. He would like to be able to get out to visit the museums in Newcastle.
A researcher visits Michael one day and shows him the *FriTab*. Michael explains to the researcher that he has mobility problems so wouldn't be able to get out much. But the researcher explains the *FriWalk* to him which is owned by several shops, galleries and museums in the area. He also explains that people on the *FriTab* network may be able to help him get transported to locations and events. So Michael enters his details into the system and tells it that he has mobility problems. The information on Michael's profile are also updated by his doctor, who also enters some contraints concerning the activities Michael can safely carry out.
The next day, the *FriTab* shows Michael that a tour is being organised at the Hancock Museum. It invites him to attend and tells him that another person attending would be willing to pick him up. The system knows that Michael enjoys natural history and that he also has mobility problems. It knows that the museum has several *FriWalk* devices that can help Michael. It also knows that one other attendee has a car and is willing to transport friends. Michael is hesitant but agrees to give it a try. So he tells the system that he will attend. The *FriTab* tells him that Jane will pick him up before the event in her car. Michael tells her his address.
At the arranged time, Jane picks Michael up and they drive to the museum. When he arrives at the museum, he is given a *FriWalk* which helps him to walk with the rest of the tour group. After the tour is over, the *FriWalk* even suggests a guided tour of its own that Michael can do alone without violating the medical prescriptions. However, Michael is tired but decides to come back and try the guided tour another day. The *FriTab* forwards the activity log file to the network for user profile updates.

Third Use Case

Dorothy is a 69 year old woman who lives alone in Blaydon. Dorothy uses a walker to get around because she finds that it gives her confidence after her fall one year ago. Moreover, it helps her in keeping a constant physical activity for a correct rehabilitation, according to the medical prescriptions. Dorothy loves shopping and likes it when her friend occasionally takes her shopping at the MetroCentre. While she likes the MetroCentre, she is nervous about going there alone and worries that she would get lost. But she would like to go there more often.
Dorothy is shown the *FriTab* and told that it clips onto *FriWalk* devices which are available at the MetroCentre. She decides to try out the *FriTab* system. Several days later, the system suggests that Dorothy visit the MetroCentre to enjoy some shopping. The system has noticed that Dorothy has stayed indoors for several days and believes that she would benefit from getting out. Dorothy thinks that it would be a good idea and asks the *FriTab* for more information. The *FriTab* suggests that she get the 2:15 bus from the nearby bus stop which will take her to the MetroCentre. It tells her that it will give her directions to the MetroCentre and will help her find her way around inside.
She gets the bus and travels to the MetroCentre. When she gets there she swaps her walker for a *FriWalk* and clips in her *FriTab*. The *FriTab* shows her that several shops have sales and gives her directions. Furthermore, it also advices her of the presence of her friends Rita and Marion, both equipped with a *FriWalk*. Dorothy meet them in front of the Central Café and then take a walk in some shops. The *FriTab* suggest them to go to the theatre insinde the mall to see a romantic movie. After the movie, Rita and Marion decided to go home, while the *FriTab* suggests Dorothy to visit the mall first floor to take a look to some very affordable items at the shoe shop. Meanwhile, Dorothy accomplishes her daily schedule of physical activities. After a while, when she starts feeling tired, she presses a button on the *FriTab* and it directs her back to where her walker is. She unclips her *FriTab* and it tells her where to get the bus home.

5 The choice of ACANTO

The Activity Planner decides a plan (Activity Plan) that implements an activity with several possibilities for its execution. The Activity Plan is refined picking the most appropriate choice by the Reactive Planner, which leverages its "on-the-ground" knowledge (Task 5.2). Therefore, the formal language for expressing plans should be flexible enough to allow for a sequence of refinements. Our starting point for this formalism will be Planning Domain Definition Language (PDDL). PDDL is widely used by researchers in the field of planning, and is the official language of the International Planning Competition. Over the years,

various extensions and improvements have been proposed to allow the definition of more complex and realistic problems and scenarios. The version 2.1 of PDDL introduces the possibility to model actions having variable durations and producing continuous effects.

The subsequent version, PDDL 3.0, is an extension providing the formalism to express constraints and preferences that must be considered by the planner for the generation of the final plan [56]. These features are particularly suitable for the ACANTO framework, in particular to model the various activities, giving different scores to each task based on the profile of the user. In addition, hard constraints can be described based on the needs of each user (for example to remain always within a given distance from the bathroom). The generated plan must respect all the hard constraints, and maximize the score based on the expressed preferences. The probabilistic extension of PDDL (PPDDL) provides a set of statements that can be used to model uncertain actions having different results, each with a given probability. The outcome of an action can affect the value of the reward. The produced plan must therefore choose a sequence of actions maximizing the probability of reaching the goal or maximizing the expected value of the reward. With respect to the scenarios depicted previously, we can report the following list of hard constraints, preferences and probabilistic preferences:

Scenario	Hard Constraints	Preferences	Probabilistic preference
Use Case I: Isabel	• Always remain within distance x from the bathroom	• Walking • Meeting someone who enjoys walking • Visiting the grocery store	• Level of tiredness always below y
Use Case II: Michael	• Avoid stairs • Maximum walked distance below x • Speed always below y • Stop every z meters and rest for some minutes	• Interest in natural history • Visiting museums	• Avoid crowded rooms
Use Case III: Dorothy	• Never remain alone • Walk at least x meters • Walk at least y minutes (medical prescriptions)	• visiting shops with sales • visiting clothing shops • watching romantic movies	• Waiting at most x minutes at the bus stop

For our purposes, there is also the need to model the coexistence of interacting people. Planning problems involving more than one person can be modelled with the adoption of a syntax like the one proposed for the multiple agents extension of the PDDL (MA-PDDL) [57]. Another language derived from PDDL that presents features which may be useful for the description of planning problems within ACANTO, is the NU-PDDL [58]. In fact, it provides a specific syntax to express temporal constraints that must be respected by the generated plan, based on Computation Tree Logic (CTL) formulae.

Another important advantage of using PDDL and its variants is that these languages are widely used by the planning community, and for this reason there exists numerous open-source applications able to produce a valid plan to solve the given problem. For example, over the years a number of planners has been developed and submitted to the various International Planning Challenges, and many of them are freely available, together with their source code. We can thus take advantage of these tools, basing our work on some of their ideas, and integrating and adapting them for our purposes. In light of the previous analysis, perfect candidates for ACANTO are tools like OPTIC (Optimising Preferences and Time-Dependent Costs) [59], a planner supporting temporal problems with preferences expressed using the PDDL3 formalism.

5.1 Planning Domain

From the persepctive of the Activity Planner, all the social activities share the same family of possible elementary actions (instantiated with different parameters), hard and soft constraints, and utility functions. Thus, the Activity Planner adopts a static PDDL domain to model all the possible activities that it will be required to refine into a sequence of Tasks, defined as follows:

```
;; DOMAIN FOR THE ACTIVITY PLANNING
;; A domain represents a general class of problems, sharing the same predicates, functions, and
;; possible actions
(define (domain Activity-Plan)

;; All the required features must be indicated, so the planner understands which features
;; it must provide
(:requirements :strips :typing :fluents :negative-preconditions :equality :durative-actions :time :timed-initial-literals)

;; From the persepctive of the Activity Planner, the plan corresponds to the visit of a sequence of Points of Interest, each associated with some metric information
(:types
    Poi - object
)

;; Here we define all the logical predicates for the problem, which can involve objects of the domain,
;; and define which properties are true at each state during the execution of the plan
(:predicates
    ;; the predicate “at” defines where an user is during each state. For example “at p0” means
    ;; that the user is currently at the Poi p0
    (at ?p - Poi)

    ;; the predicate “visitable” defines which Pois are visitable (for example some
    ;; activities could be temporarily not available
    (visitable ?p – Poi)

    ;; the predicate “to-visit” defines which Pois have still to be visited
    (to-visit ?p – Poi)

    ;; the predicate “link” is used to model the map of the environment, defining which Pois are
    ;; directly connected.
    (link ?p1 ?p2 - Poi)

    ;; the predicate “idle” defines whether the user is currently not executing any action
    (idle)
  )

;; Here we define all the functions for the problem. Functions can take zero or more arguments,
;; defining which objects of the domain are involved. To each combination of arguments, a
;; function can associate a numerical value, which can change over time.
;; For example there could be a function “age ?p – Person”, defining for each person its age.
  (:functions
    ;; the function “min-total-distance” keeps track of the minimum distance walked from
    ;; the beginning up to now
    (min-total-distance)

    ;; the function “max-total-distance” keeps track of the maximum distance walked from
    ;; the beginning up to now
    (max-total-distance)
```

```
 ;; the function "min-partial-distance" keeps track of the minimum distance walked from
 ;; the last rest up to now (it is reset to 0 during each rest)
 (min-partial-distance)

 ;; the function "max-partial-distance" keeps track of the maximum distance walked from
 ;; the last rest up to now (it is reset to 0 during each rest)
 (max-partial-distance)


 ;; the function "satisfaction" is used to model the current level of satisfaction for the user,
 ;; which depends on the scores associated to the visited Pois
 (satisfaction)

 ;; the functions "min-distance" and "max-distance" are used to model the minimum and
 ;; maximum distance between each pair of connected Pois. In this way, it is possible to
 ;; compute the total walked distance.
 (min-distance ?p1 ?p2 - Poi)
 (max-distance ?p1 ?p2 - Poi)

 ;; the function "rest-duration" indicates how long each rest should last
 (rest-duration)

 ;; the function "speed" is used to model the speed at which Michael should move around
 (speed)

 ;; the function "visit-time" indicates for each Poi ?p how much time it is required to visit it
 (visit-time ?p – Poi)

 ;; the function "interest" indicates for each Poi how much the user is interested in it
 (interest ?p – Poi)
)

;; durative-actions are used to model actions with some (even variable) duration, and having
;; temporal conditions and effects

;; the durative-action move represents  the task of moving from one Poi of the museum
;; to another
(:durative-action move
 :parameters (?from – Poi ?to – Poi)

 ;; the duration of the move action depends both on the distance between the two Pois, and also
 ;; on the speed at which the user moves
 :duration (= ?duration (/ (max-distance ?from ?to) (speed)))

;; the conditions are that the user at the beginning must be at the ?from Poi, he
;; must be idle, and the two Pois must be connected
 :condition
   (and
    (at start (at ?from))
    (at start (link ?from ?to))
    (at start (idle))
```

```
    )

   ;; the effects are that at the end the user will be at the destination (?to) Poi, and both the
   ;; partial and the total walked distance are suitably increased
   :effect
    (and
     (at start (not (at ?from)))
     (at start (not (idle)))
     (at end (at ?to))
     (at end (increase (min-total-distance) (min-distance ?from ?to)))
     (at end (increase (max-total-distance) (max-distance ?from ?to)))
     (at end (increase (min-partial-distance) (min-distance ?from ?to)))
     (at end (increase (max-partial-distance) (max-distance ?from ?to)))
     (at end (idle))
    )
  )

  ;; the durative-action "rest" represents  the task of resting for some time
  (:durative-action rest
   :parameters (?p - Poi)

   ;; the duration of the rest is indicated by the value of the "rest-duration" function
   :duration (= ?duration (rest-duration))

   ;; At the beginning, the user must be idle
   :condition
    (and
     (at start (idle)
     (at start (at ?p))
     (over all (at ?p))

   ;; at the end the user exits from the "resting" state, and the partial walked distance is reset to 0
   :effect
    (and
     (at start (not (idle)))
     (at end (idle))
     (at end (assign (partial-distance) 0))
    )
  )

;; the durative-action "visit" represents  the task of visiting a Poi
(:durative-action visit
   :parameters (?p – Poi)

   ;; the duration depends on the specific "visit-time" for Poi ?p
   :duration (= ?duration (visit-time ?p))

   ;; the Poi must not have been already visited, and the user must be idle.
   ;; In addition,  the user be at that specific Poi and the Poi
   ;; must be in the state "visitable"
```

```
    :condition
     (and
      (at start (not (visited ?p)))
      (at start (idle))
      (at start (visitable ?p))
      (over all (visitable ?p))
      (at start (at ?r))
      (over all (at ?r))
     )

    ;; At the end the Poi is flagged as "visited", and the level of satisfaction is increased
    ;; depending on the interest for the specific Poi
    :effect
     (and
      (at start (not (idle)))
      (at end (idle))
      (at end (visited ?r))
      (at end (increase (satisfaction) (interest ?p)))
     )
    )
)
```

5.2 Extension to groups

Up to now we have shown the main concepts and ideas adopted to model Activity Plans, considering a scenario with a single user. However, in order to achieve the social aspect of ACANTO activities, the planner must support a context where an activity is carried out by more than one person.

However, since the kind of possible actions, constraints and preferences remains the same, a minor extension of the original model suffices to provide support for a multi-agent context.

The planning domain can be extended by introducing a new type of object to represent an Agent named "User". In addition, a score is given by each User to each PoI. Soft constraints associated to each person (e.g. "prefer plans avoiding the use of lifts") are modelled as costs given to the different PoIs and connectors between pairs of PoIs:

```
;; the level of interest of an user ?u to a specific PoI ?p
;; (inferred from the user profile)
(interest ?u - User ?p - Poi)

;; the cost given by an user ?u to a specific PoI ?p
;; (inferred from the user profile)
(poi_cost ?u - User ?p - Poi)

;; the cost given by an user ?u to a link between two PoIs
;; ?p1 and ?p2 (inferred from the user profile)
(link_cost ?u - User ?p1 - Poi ?p2 - Poi)
```

The hard constraints are generated as the conjunction of all the hard constraints of each user. For example, the constraint on the total duration *max-global-time* is determined as:

$$\text{max-global-time} = min_{u \in \text{Users}} \text{max-time-allowed}\ (u)$$

The metric function to optimize is then the weighted sum of the overall score of the visited PoIs for all the users, and the overall costs deriving from the violation of the soft-constraints:

$$metric = w_1 \sum_{p \in Pois} \sum_{u \in Users} x[i]\, interest(u,p) \;-\; w_2 \sum_{p \in Pois} \sum_{u \in Users} x[i]\, poi\_cost(u,p) \;-\; w_3 \sum_{\langle p_1,p_2\rangle \in Links} \sum_{u \in Users} y[\langle p_1,p_2\rangle]\, link\_cost(p_1,p_2)$$

where x[i] is a Boolean indicator variable, indicating whether the i-th PoI is part of the solution, $y[\langle p_i, p_j\rangle]$ indicates wheter the link $\langle p_i, p_j\rangle$ is part of the solution, and $w_1, w_2, w_3 \in \mathbb{R}$ are weighting factors, allowing us to give different weights to the overall interest and to the violation of soft constraints involving both PoIs and connectors.

5.3 An example

In what follows we present how the Use Case II (referred to Michael) can be encoded into an instance of a PDDL planning problem (according to the Activity Planning domain), that can be solved by the Activity Planner to determine the plan to be executed.

```
;; PROBLEM VISIT
;; a problem represents a specific instance of a planning problem, defining which are the specific
;; objects, and the initial and goal conditions
(define (problem VISIT)
   (:domain MUSEUM)

   ;; for this scenario, we model the different Pois for the Museum of National History.
   ;; In particular, there is a room dedicated to plants, one dedicated to animals and one dedicated to
   ;; fungi, all represented as Points of interest. In addition, the atrium is represented as a Poi
   ;; where the execution of the plan must start and end
   (:objects
    roomPlants - Poi
    roomAnimals - Poi
    roomFungi – Poi
    …
    atrium – Poi
)

   (:init
    ;; initially Michael is in the atrium. We define the links between the various rooms.
    ;; Then we define which rooms are “visitable”.
    (at atrium)
    (link atrium roomPlants)

    (link roomPlants atrium)
    (link roomPlants roomAnimals)
    (link roomPlants roomFungi)
    …

    (visitable roomPlants)
    (visitable roomAnimals)
```

```
    (visitable roomFungi)

    ;; here we initialise the values of the functions, setting to 0 the walked distances, and defining
    ;; Michael's level of interest for each of the Pois (30 for plants, 60 for animals, and 15 for fungi).
    ;; We also define the distance between each pair of connected locations, the duration of each rest,
    ;; the walking speed, and the time required to visit each room.
    (= (min-total-distance) 0)
    (= (max-total-distance) 0)
    (= (min-partial-distance) 0)
    (= (max-partial-distance) 0)

    (= (interest roomPlants) 30)
    (= (interest roomAnimals) 60)
    (= (interest roomFungi) 15)
    (= (max-distance atrium roomPlants) 15)
    (= (max-distance roomPlants roomAnimals) 10)
    …
   (= (rest-duration) 300)

   (= (speed) 1)
   (= (visit-time roomPlants) 600)
   (= (visit-time roomAnimals) 900)
   (= (visit-time roomFungi) 600)

   ;; here we define some timed initial literals, to model predicates which become true/false at some
   ;; specific time instants after the execution of the plan, and which can be used to model some
   ;; known exogenous events.
   ;; For example, here we model the fact that between time 700 and time 1000 the room
   ;; "roomPlants" becomes unavailable (e.g. there is a guided tour during that interval)
   (at 700.00 (not (visitable roomPlants)))
   (at 1000.00 (visitable roomPlants))
)
;; for this problem, Michael's final goal is to be at the atrium and to have walked a distance between
;; 30 and 100 meters
(:goal
   (and
    (at atrium)
    (>= (min-total-distance) 30)
    (<= (max-total-distance) 100)
    …
   )
)

;; here we define some hard constraints and some preferences
(:constraints
   (and
    ;; the partial distance walked (between two consecutives rest) must always be lower than 25
    ;; meters
    (always (<= (partial-distance) 25))

;; for this problem, we want to maximize Michael's satisfaction
(:metric maximize
```

```
    (satisfaction)
)
```

6 Semantic representation of plans

In ACANTO, the notion of optimality is with respect to the goals of the activity, which may require a *minimum* distance travelled, a maximum amount of time or a minimum number of calories burned. While the notion of undesirable circumstances still exists (e.g., undesirable areas in the environment and crowding), maximising progress, in the sense of minimising distance and time spent travelling, may not be the primary goal. Users or carers may nevertheless specify a maximum time for an activity. In ACANTO, the user may be part of a group, thus the execution of an activity, the constraints and metrics to apply to the various point of interest and interconnections must be considered with respect to the entire group.

From the perspective of the Activity Planner, an activity corresponds to a sequence of points of interest to visit in a certain order, depending on the existing interconnections among the various Pois in the abstract representation of the environment. The Activity Planner is thus required to synthesize a sequence of tasks representing the transfer and visit of the different Pois, trying to optimise the overall score (depending both on the level of interest for the different Pois, and on the satisfaction of the soft constraints) while fulfilling all the hard constraints. The tasks composing a plan are thus of the form "*nextPoi* P_i P_j". To each of these tasks is associated a sequence of elementary actions to be performed before leaving the current Poi P_i, the action corresponding to the motion between P_i and P_j, and a sequence of actions to be executed after reaching P_j. The actions to be performed before leaving or after reaching a certain point of interest depend on the specific point of interest involved (e.g. if the Poi is a painting in a museum, the action when reaching it could be to display some related information on the FriTab).

7 Monitoring the execution

At an abstract semantic level, an activity is the traversal of a graph and an activity plan is a concatenation of edges. At a concrete syntactic level, such a path will be represented by elementary control actions in a language such as PDDL, as described above. The control actions will specify precisely how to navigate from one node in the environment to the next. To meet the high level requirements of an activity, such as the number of calories to consume, the control actions may also specify a required speed and other information, such as probabilistic distributions. The Activity Planner makes use of probabilistic information (e.g., the probability of being able to make progress at a desired speed, given the amount of crowding) in a deterministic way.

Each control action has inherent pre- and post-conditions that may or must be satisfied, depending on the context. Many of the pre- and post-conditions of control actions will be implicitly encoded in the edge weighting function. A priori, the plan will be designed to satisfy all constraints, but the user may from the outset be forced or choose to deviate from the original plan. Such deviations may arise by virtue of crowding or due to the uncooperativeness of the user. In the first case, the deviation could be suggested by the Reactive Planner; in the second case, the user may simply ignore the instructions given by the *FriTab*. In either case, there may be a significant reduction in the probability of satisfying the goals of the original activity plan. It might also be the case that, in deviating, the user performs more exercise and burns more calories than anticipated. It is therefore necessary to monitor the plan's progress and react accordingly.

The Activity Monitor will thus construct path metrics that quantify what has actually taken place so far (e.g., what distance and how fast the user has travelled; the number of calories the user has consumed; social interaction) and what remains to be done to achieve the goals of the activity (remaining distance to travel and number of calories yet to burn, etc). The metrics can be represented as edge weights that are summed over the length of a path. In the case that the Activity Monitor judges that the current plan is unlikely to achieve the goals of the activity, it will trigger a re-plan.

In DALi, a new long term plan is generated if the user strays too far from the original plan. In ACANTO, the initially proposed route of an activity may be considered only a "soft" constraint that may be violated, with the metrics over the route being more important. In contrast, constraints that arise from the user's profile, such as the user's maximum speed and the maximum time the user is prepared to spend on the activity, will be considered "hard" constraints that must not be violated.

A new activity plan will therefore be generated when the activity metrics diverge significantly from their desired values or when the Activity Monitor predicts that the metrics will diverge in the future with high probability. The Activity Planner may skip some or all of the remaining planned sub-activities. This could arise because the allotted time for the activity has been reached or because the user has already had sufficient exercise or consumed sufficient calories as a result of unplanned detours. In general, the Activity Planner will try to achieve as many of the soft constraints and goals as possible, while always respecting the hard constraints.

The Activity Planner may have to consider conflicting constraints that cannot be entirely encoded in an edge weighting function. For example, circumstances could arise such that the remaining time allotted to an activity is insufficient for the user to consume the required number of calories or travel the required distance, given the users maximum comfortable speed. This will become clear from the output of the path planner (e.g., the chosen optimal path will have an overall time that is too great), but must be handled by higher level logic. To resolve such problems, we assume that activities will be composed from a series of sub-activities (tasks), each making a quantified and prioritised contribution to the overall activity. The Activity Planner will thus be able to discard the least beneficial remaining sub-activities, in order to achieve as much of the originally planned benefit as possible, without violating the user's hard constraints.

The foregoing descriptions have been user-centric, but it is the specific intention of ACANTO to consider group activities. There must be some online collaboration between the motion planners of different users, to maintain the cohesion of the group.

The means by which the Reactive Planner will maintain group cohesion on a specified plan has been described in Deliverable D5.2, however the Activity Planner may wish to change the plan with respect to an individual's specific requirements. In general, it is desirable to make each *FriWalk* / *FriTab* as autonomous as possible. To prevent the Activity Planner compromising this goal (e.g., by needing to regularly communicate with other users in order to find a consensus plan), we suppose that the system will pre-define how the activity will be modified in the event that it must be truncated. To implement this "implicit collaboration" efficiently, each Activity Planner should be aware of the limitations of the other users. A further problem is that although the participants of group activities will be chosen to have roughly equivalent abilities, the actual distance travelled (effort expended, calories burned, etc) by individuals will be different. This limits the accuracy with which it is possible for one *FriWalk* / *FriTab* to predict the re-planning needs of another without sharing information.

The a priori information used by the Activity Planner and the Monitor Service is based on statistical data collected from previous instances of ACANTO Activities recorded in the past by the Activity evaluator or from simulation and predictions. Additional live information on the environment can be provided by the real time sensing system.

8 Conclusion

The PDDL language is suitable to represent activity plans at a syntactic level. We have shown that this language is adequately expressive to represent probabilistic information and other annotations necessary for the activity planning required by the ACANTO case studies. We have described how we intend to abstract the environment into a graph data structure and how various metrics relating to an activity can be encoded in the weights assigned to its edges. Paths in the environment that optimise parameters of an activity can thus be found by finding the optimal path in the abstract graph. The metrics will also be used to monitor the progress of the user with respect to an activity and anticipate when the activity needs to be re-planned or terminated.

10 Bibliography

[1] ACANTO Consortium. ACANTO Project, Feb. 2015.

[2] DALi Consortium. DALi Project, Nov. 2011.

[3] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[4] J. C. Doyle, B. A. Francis, and A. Tannenbaum. *Feedback control theory, volume 1*. Macmillan Publishing Company, New York, 1992.

[5] D. E. Koditschek and E. Rimon. *Robot navigation functions on manifolds with boundary*. Advances in Applied Mathematics, 11(4): 412–442, 1990.

[6] S. R. Lindemann, I. I. Hussein, and S. M. LaValle. *Real time feedback control for nonholonomic mobile robots with obstacles*. In Proc. IEEE Conference on Decision and Control, pp. 2406–2411, 2006.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. IEEE Transactions on Robotics and Automation, 12(4): 566–580, 1996.

[8] S. M. LaValle. *RRT-connect: An efficient approach to single-query path planning.* In Proc. IEEE Conference on Robotics and Automation, pp. 995-1001, 2000.

[9] S. Karaman and E. Frazzoli. *Sampling-based algorithms for optimal motion planning*. The International Journal of Robotics Research, 30(7): 846–894, 2011.

[10] M. Ghallab, D. Nau, and P. Traverso. *Automated planning: theory & practice*. Elsevier, 2004.

[11] T. L. Dean and M. P. Wellman. *Planning and control*. Morgan Kaufmann Publishers Inc., 1991.

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

[13] M. Fox and D. Long. *PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains*. Journal of Artificial Intelligence, v. 20, pp. 61-124, 2003.

[14] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT press, 1999.

[15] C. Baier, J. P. Katoen, et al. *Principles of model checking*. MIT press Cambridge, 2008.

[16] K. L. McMillan. *Symbolic model checking*. Springer, 1993.

[17] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. *Symbolic planning and control of robot motion [grand challenges of robotics]*. Robotics & Automation Magazine, IEEE, 14(1): 61–70, 2007.

[18] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. *Temporal logic motion planning for dynamic robots*. Automatica, 45(2): 343–352, 2009.

[19] A. Bhatia, L. E. Kavraki, and M. Y. Vardi. *Sampling-based motion planning with temporal goals*. In Proc. IEEE International Conference on Robotics and Automation, pp. 2689–2696, 2010.

[20] A. Colombo, D. Fontanelli, A. Legay, L. Palopoli and S. Sedwards. *Efficient customisable dynamic motion planning for assistive robots in complex human environments*. Journal of Ambient Intelligence and Smart Environments, 7(5): 617-633, IOS PRESS NIEUWE HEMWEG 6B, 1013 BG AMSTERDAM, NETHERLANDS, 2015.

[21] P. Tabuada and G. J. Pappas. *Linear time logic control of discretetime linear systems*. IEEE Transactions on Automatic Control, 51(12): 1862–1877, 2006.

[22] E. Aydin Gol and M. Lazar. *Temporal logic model predictive control for discrete-time systems*. In Proceedings of the 16th international conference on Hybrid systems: Computation and Control, 343–352. ACM, 2013.

[23] B. Yordanov and C. Belta. *Formal analysis of discrete-time piecewise affine systems*. IEEE Transactions on Automatic Control, 55(12): 2834–2840, 2010.

[24] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta. *Temporal logic control of discrete-time piecewise affine systems*. IEEE Transactions on Automatic Control, 57(6): 1491–1504, 2012.

[25] G. E. Fainekos and G. J. Pappas. *Robustness of temporal logic specifications for continuous-time signals*. Theoretical Computer Science, 410(42): 4262–4291, 2009.

[26] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta. *Probabilistic temporal logic falsification of cyber-physical systems*. ACM Transactions on Embedded Computing Systems (TECS), 12(2s): 95, 2013.

[27] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. *Temporal logic motion planning for dynamic robots*. Automatica, 45(2): 343–352, 2009.

[28] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. *Mdp optimal control under temporal logic constraints*. In Proc. IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), 532–538, 2011.

[29] E. M. Wolff, U. Topcu, and R. M. Murray. *Robust control of uncertain markov decision processes with temporal logic specifications*. In Proc. IEEE Conference on Decision and Control, pp. 3372–3379, 2012.

[30] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. *Temporal-logic-based reactive mission and motion planning*. IEEE Transactions on Robotics, 25(6): 1370–1381, 2009.

[31] T. Wongpiromsarn, U. Topcu, and R. M. Murray. *Receding horizon control for temporal logic specifications*. In Proceedings of the 13th ACM international conference on Hybrid systems: computation and control, 101–110, 2010.

[32] M. Guo, K. H. Johansson, and D. V. Dimarogonas. *Revising motion planning under linear temporal logic specifications in partially known workspaces*. In IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 2013.

[33] S. C. Livingston, R. M. Murray, and J. W. Burdick. *Backtracking temporal logic synthesis for uncertain environments*. In Proc. IEEE International Conference on Robotics and Automation, pp. 5163–5170, 2012.

[34] G. E. Fainekos. *Revising temporal logic specifications for motion planning*. In Proc. IEEE International Conference on Robotics and Automation, pp. 40–45, 2011.

[35] K. Kim and G. E. Fainekos. Approximate solutions for the minimal revision problem of specification automata. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 265–271, 2012.

[36] V. Raman and H. Kress-Gazit. *Analyzing unsynthesizable specifications for high-level robot behavior using ltlmop*. In Computer Aided Verification, 663–668. Springer, 2011.

[37] M. Guo and D. V. Dimarogonas. *Reconfiguration in motion planning of single- and multi-agent systems under infeasible local LTL specifications*. In Proc. IEEE Conference on Decision and Control, 2013.

[38] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus. *Least-violating control strategy synthesis with safety rules*. In Proceedings of the 16th international conference on Hybrid systems: computation and control, 1–10, 2013.

[39] J. Tumova, L.I.R. Castro, S. Karaman, E. Frazzoli and D. Rus. *Minimum-violation LTL planning with conflicting specifications*. In American Control Conference (ACC), 2013, pp.200-205, 17-19 June 2013.

[40] E. H. Durfee. *Distributed problem solving and planning*. In Multi-agent systems and applications, 118–149. Springer, 2006.

[41] J. R. Kok and N. Vlassis. *Collaborative multiagent reinforcement learning by payoff propagation*. The Journal of Machine Learning Research, 7: 1789–1828, 2006.

[42] M. Kloetzer and C. Belta. *Automatic deployment of distributed teams of robots from temporal logic motion specifications*. IEEE Transactions on Robotics, 26(1): 48–61, 2010.

[43] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta. *Automatic deployment of robotic teams*. Robotics & Automation Magazine, 18(3): 75–86, 2011.

[44] A. Ulusoy, S. L. Smith, X. C. Ding, and C. Belta. *Robust multi-robot optimal path planning with temporal logic constraints*. In Proc. IEEE International Conference on Robotics and Automation, pp. 4693–4698, 2012.

[45] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos. Decentralized multi-agent control from local ltl specifications. In Proc. IEEE International Conference on Decision and Control, pp. 6235–6240, 2012.

[46] K. Kim and G. Fainekos. *Revision of specification automata under quantitative preferences*. In Proc. IEEE International Conference on Robotics and Automation, pp. 5339-5344, May 31 2014-June 7 2014.

[47] E. P. Pednault. *ADL: Exploring the middle ground between STRIPS and the situation calculus*. In Proceedings of the first international conference on Principles of knowledge representation and reasoning, pp. 324-332, 1989.

[48] J. Hoffmann, and B. Nebel. *The FF planning system: Fast plan generation through heuristic search*. Journal of Artificial Intelligence Research, pp. 253-302, 2001.

[49] M. Ghallab, and H. Laruelle. *Representation and control in IxTeT, a temporal planner*. In Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94), pp. 61–67, 1994.

[50] *NDDL Reference manual*. Available from https://github.com/nasa/europa/wiki/NDDL-Reference, last visit: 24-01-2016

[51] D.E. Smith, J. Frank, and W. Cushing. *The ANML language*. In The ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), 2008.

[52] J. Hoffmann, and R. Brafman. *Contingent planning via heuristic forward search with implicit belief states*. Proc. ICAPS. Vol. 2005, 2005.

[53] C. Muise, V. Belle, and S. A. McIlraith. *Computing contingent plans via fully observable non-deterministic planning*. The 28th AAAI Conference on Artificial Intelligence, 2014.

[54] H. L. Younes, H. L., and M. L. Littman. *PPDDL1.0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects*. In Proceedings of the 14th International Conference on Automated Planning and Scheduling, 2004.

[55] S. Sanner. *Relational Dynamic Influence Diagram Language (RDDL): Language Description*. Unpublished Paper, 2010. Available from http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/, last visit: 25-01-2016.

[56] A. Gerevini, and D. Long. *Plan constraints and preferences in PDDL3. The Language of the Fifth International Planning Competition*. Technical Report, Department of Electronics for Automation, University of Brescia, Italy, 75, 2005.

[57] D. L. Kovacs. *A Multi-Agent Extension of PDDL3.1*. In Proceedings of the 3rd Workshop on the International Planning Competition, 2012.

[58] *NuPDDL: nondeterminism and more in PDDL*. Available from http://mbp.fbk.eu/NuPDDL.html, last visit: 25-01-2016

[59] *OPTIC: Optimising Preferences and Time-Dependent Costs.* Available from http://www.inf.kcl.ac.uk/research/groups/PLANNING/, last visit: 28-01-2016

[60] R. Finkel and J. Bentley. *Quad trees a data structure for retrieval on composite keys*, Acta Informatica, Springer-Verlag, pp. 1–9, 1974.