On the Manipulation of Articulated Objects in Human-Robot Cooperation Scenarios

Alessio Capitanelli, Marco Maratea, Fulvio Mastrogiovanni[1], Mauro Vallati

A. Capitanelli, M. Maratea and F. Mastrogiovanni are with the Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genoa, via Opera Pia 13, 16145 Genoa, Italy.
M. Vallati is with the School of Computing and Engineering, University of Huddersfield, Queensgate, HD13DH, Huddersfield, United Kingdom.

Abstract

Articulated and flexible objects constitute a challenge for robot manipulation tasks, but are present in different real-world settings, including home and industrial environments. Approaches to the manipulation of such objects employ *ad hoc* strategies to sequence and perform actions on them depending on their physical or geometrical features, and on *a priori* target object configurations, whereas principled strategies to sequence basic manipulation actions for these objects have not been fully explored in the literature.

In this paper, we propose a novel action planning and execution framework for the manipulation of articulated objects, which (i) employs action planning to sequence a set of actions leading to a target articulated object configuration, and (ii) allows humans to collaboratively carry out the plan with the robot, also interrupting its execution if needed.

The framework adopts a formally defined representation of articulated objects. A link exists between the way articulated objects are perceived by the robot, how they are formally represented in the action planning and execution framework, and the complexity of the planning process.

Results related to planning performance, and examples with a Baxter dual-arm manipulator operating on articulated objects with humans are shown.

[1]Corresponding author. *Email address*: `fulvio.mastrogiovanni@unige.it`

Keywords:
Planning; Knowledge representation; Software architecture; Articulated object.

1. Introduction

The Industry 4.0 paradigm is expected to redefine the nature of shop-floor environments, including the role played by robots in the manufacturing process [1, 2]. One of its main tenets is the increased customer satisfaction via product personalization and just-in-time delivery. A higher level of flexibility in manufacturing processes is needed to cope with diversified demands, especially in low-automation tasks. Collaborative robots are regarded as a valuable aid to shop-floor operators, who can supervise robots' work and intervene when needed [3], whereas robots can be tasked with difficult or stressful operations.

Human-robot cooperation (HRC) in the shop-floor is a specific form of human-robot interaction (HRI) with two important specificities. The first is the fact that cooperation is targeted to a well-defined objective (e.g., an assemblage, a unit test, a cable harnessing operation), which must be typically achieved in a short amount of time. The second has to do with the fact that humans need to feel in control [4, 5]: human behaviour could be unpredictable in specific cases, with obvious concerns about safety [6, 7], humans may not fully understand robot goals [8], and robot actions may not be considered appropriate for the peculiar cooperation objectives [9, 5].

As far as the cooperation process is concerned, two high-level directives must be considered:

- D_1 it is necessary to adopt human-robot *cooperation models* and the associated robot action planning techniques to meet cooperation objectives [10, 11];
- D_2 robots must be flexible enough to adapt to human actions while (i) fulfilling the overall cooperation objectives [12, 13], and (ii) to make their intentions clear to human operators [14, 15].

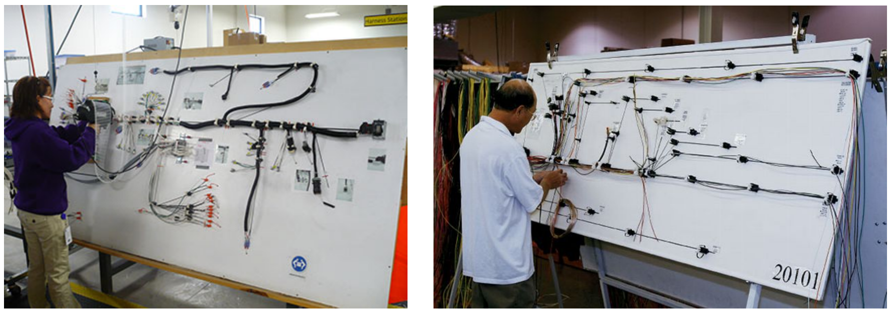

Figure 1: Two examples of a cable harnessing operation.

These directives lead to three functional requirements for a HRC architecture. Collaborative robots must be able to:

R_1 recognize the effects of human operator actions [16];

R_2 adapt their behaviour considering human actions and the whole cooperation objectives;

R_3 employ planning techniques allowing for an appropriate action re-planning when needed, e.g., when planned actions cannot be executed for sudden changes in the environment or inaccurate modelling assumptions [17].

Among typically shop-floor tasks, the manipulation of articulated or flexible objects, e.g., cable harnessing operations, is particularly challenging [18, 19, 20, 21], as can be seen in Figure 1. In this example, it is required to plan the expected cable configurations on the harnessing table *in advance*, thus confirming requirement R_3. Furthermore, it is necessary to keep a cable firm using more than two grasping points and to re-route the wiring pattern, which – when done collaboratively with a robot, for instance to place bundle retainers or junction fixtures – leads to requirements R_1 and R_2 above.

In the literature, the problem of determining the 2D or 3D configuration of articulated or flexible objects has received much attention in the past few years [22, 23], whereas the problem of obtaining a target configuration via manipulation has been explored in motion planning [24, 25, 26]. In the context of HRC, perception and manipulation are only part of the challenges to address.

Conceptually, the outcome of such approaches is a *continuous mapping* from an initial to a target object's configuration [27, 28, 25, 29], subject to simplifying hypotheses related to object models [30, 31, 32, 33, 34]. This remark leads to two further requirements. Collaborative robots must be able to:

R_4 adopt a representation to be used by action planners, and segment the whole manipulation problem in simpler actions, each action leading to a new intermediate configuration;

R_5 represent actions using a formalism allowing for plan executions that are robust with respect to unexpected events (e.g., the human operator suddenly intervenes), and modelling errors (e.g., not modelled objects to be removed from the workspace).

In this paper, we focus on the human-robot cooperative manipulation of articulated objects [24], and we contribute to the literature as follows: (i) the design and development of two representation and planning models for the classification of articulated object configurations and the sequencing of manipulation actions; to that aim, we build an OWL-DL ontology to represent articulated objects and the actions on them, and we use Planning Domain Definition Language (PDDL) to define such planning models [35]; we employ two state-of-the-art PDDL planners, namely Probe [36] and Madagascar [37] as well as the VAL plan validator [38], to generate manipulation plans using such models; to the best of our knowledge, such a modelling approach for articulated objects has no comparison with existing literature; (ii) the design and development of a novel reactive/deliberative architecture for HRC, which we call PLANHRC, allowing human operators to intervene as they wish during the cooperation process, and implemented on top of the ROSPlan [39] and MoveIt! [40] frameworks, thereby extending ROSPlan to HRC scenarios; and (iii) a discussion about how robot perception and object representation impact on action planning and execution in HRC scenarios, which is peculiar for the use case we consider. The PLANHRC architecture has been validated on a dual-arm Baxter manipulator.

The paper is organised as follows. Section 2 discusses relevant approaches in the literature. Section 3 introduces more formally the problem and the scenario we consider. The planHRC's architecture is described in Section 4, where the overall information flow, the representation and reasoning processes, and the planning models are discussed. Experiments to validate the architecture are described in Section 5. Conclusions follow.

2. Background

A number of studies have been conducted to investigate the role and the acceptability of automated planning techniques in HRC scenarios. Gombolay and colleagues highlight two factors as important to maximise human satisfaction in HRC [41]: (i) humans should be allowed to choose their own tasks freely, i.e., not assigned by an algorithm, subject to the fact that the cooperation is successful; (ii) the overall human-robot team's performance must be at high standards. These two factors may conflict in case of a *lazy* or *not focused* human operator's attitude. However, when required to trade-off between them, humans show a strong preference for team's performance over their own freedom. This study well fits with the requirements R_1, R_2 and R_3 outlined above, and opens up to an idea of collaborative robots as devices not only able to *aid* human workers, but also capable of *keeping them in focus* and steering the cooperation towards its objectives if deviations occur.

As a follow-up of the work discussed in [41], a study about the actual amount of control a human operator would like to have when collaborating with a robot has been reported in [42]: human workers tend not to prefer a total control of the cooperation process, rather they opt for partial control. This is confirmed by the fact that the overall team's performance seems higher when the robot determines what actions must be carried out by the human operator. A key factor for the acceptance of collaborative robots is finding a sensible – yet efficient – trade-off between performance and human control.

In order to determine such trade-off one possibility is to encode human operator preferences in the planning process [43]. In a first series of experiments, the use of human preferences in the planning algorithm led to an overall *decrease* in performance, correlated with human subjective perception of robots not in line with the main cooperation objectives. This suggests that a subjective assessment of the HRC process tends to attribute major inefficiencies to robots, and confirms that this is a crucial aspect for the applicability of collaborative robots in industrial scenarios. Techniques for HRC available in the literature target these issues only to a partial extent, and in limited contexts. In particular, it is possible to identify two relevant activity trends our approach is related to.

Approaches in the first class aim at defining cooperation models, i.e., data structures modelling the task to be jointly carried out, and algorithms operating on such data structures for the cooperation process to unfold, while keeping flexibility and human preferences into account [44, 45, 3, 10, 15, 46, 11].

A probabilistic planner is used in [44] to sequence available partial plans, which include indications about human preferred actions. Once determined, the sequence of partial plans cannot be changed, therefore no flexibility for the human is allowed. Such a limitation is partially overcome by the approach described in [45], where an algorithm to adapt on-line both the action sequence and the number of action parameters is described. This is achieved using a temporal formulation making use of preferences among actions, and using optimization techniques to identify the sequence best coping with preferences and constraints. The algorithm weighs more plan optimality (in terms of a reduced number of actions, or the time to complete the plan), and uses human preferences as soft constraints. The approach by Tsarouchi and colleagues [3] assumes that a human and a robot co-worker operate in different workspaces. The focus is on allocating tasks to the human or the robot depending on their preferences, suitability and availability, and the cooperation model is represented using an AND/OR graph. Although human preferences are taken into account, task allocation is *a priori* fixed and cannot be changed at run-time. A similar approach is considered in [10], where the assumption about the separate workspaces is relaxed. Hierarchical Task Models (HTMs) are used in [15], where the robot is given control on task allocation and execution is modelled using Partially

Observable Markov Decision Processes (POMDPs). However, the focus of this approach is on robot communication actions to enhance *trust* in the human counterpart and to share a mutual understanding about the cooperation objectives. A similar approach is adopted in [46], where HTMs are substituted by Hierarchical Agent-based Task Planners (HATPs) and POMDPs are replaced by Petri Network Plans (PNPs). However, differently from the approach in [15], the work by Sebastiani and colleagues support on-line changes during plan execution. Finally, the work by Darvish and colleagues represents cooperation models using AND/OR graphs, and allows for a switch among different cooperation sequences at runtime [11], therefore allowing humans to redefine the sequence of tasks among a predefined set of choices. The human operator does not have to explicitly signal the switch to the robot, whereas the robot adapts to the new cooperation context reactively.

The second class includes techniques focused on understanding, anticipating or learning human behaviours on-line [47, 48, 49, 50, 51, 52].

The work by Agostini and colleagues adopts classical planning approaches to determine an appropriate sequence of actions, given a model of the cooperation defined as a domain and a specific problem to solve [47]. At runtime, the system ranks a predefined series of cause-effect events, e.g., observing their frequency as outcomes of human activities, and updates the cooperation model accordingly. Markov Decision Processes (MDPs) are used in [48] to model the cooperation. In particular, the human and the robot are part of a Markov decision game, and must cooperatively conclude the game, i.e., carrying out the cooperation process. Human actions are detected on-line, which influences robot's behaviour at run-time. A similar approach, which takes into account temporal constraints among tasks, is discussed in [49]. Statistical techniques to recognise human actions and to adapt an already available plan accordingly are presented in [50]. Human deviations from the plan are detected. When this happens, re-planning (including task allocation) occurs to achieve the cooperation objectives. While the approaches discussed so far are quite conservative as far as robot's autonomy in the cooperation process is concerned, the work discussed in [51] exploits Bayesian networks to predict the occurrence and the timing of human actions. Such a prediction is used to perform preparatory actions before an event even occurs. While the overall system's performance is greatly improved, humans tend to be confused by the seemingly anticipative robot's behaviour. Hierarchical Task Networks (HTNs) are used in [52] to embed communication actions in the cooperation process. When specific deviations from the plan are detected, such communication actions enforce the adherence to the plan.

From this analysis of the literature, it is possible to frame our contributions and the main features of PLANHRC with respect to existing literature: (i) while the majority of cooperation models described in the literature do not allow human operators to decide what actions to carry out, or they do so only to a very limited extent, PLANHRC foresees a cooperation process informed by optimality in the planning process (therefore adhering to the first findings of Gombolay and colleagues), but allows humans to intervene freely, up to the limit situation where all the plan is executed by the human operator as he or she wishes (i.e., humans are given partial or total control); (ii) in PLANHRC the robot does not have to explicitly recognise human operator actions, as it is prescribed by approaches in the literature, but it focuses on their effects in the planning model, and treats any perturbation as violations with respect to the normal plan unfolding. In particular, PLANHRC takes inspiration from the findings in [41, 42, 43] to devise a cooperation model and an interaction model with the human operator with the following characteristics:

- similarly to the work in [47], the robot plans an appropriate, *optimal*, sequence of actions to determine relevant intermediate configurations for an articulated object (considered as a simplified model for a flexible object like a cable), in order to determine a final target configuration, therefore coping with requirement R_4;
- during plan execution, the robot always monitors the outcome of each action, and compares it with the target configuration to achieve, therefore limiting the burden on the human side [42];

- normally, the human can supervise robot actions: when a robot action is not successful, or a plan cannot be found, the human can intervene on the robot's behalf performing her/his preferred action sequence [43], therefore meeting R_1 and R_2;

- at any time, the human can intervene (e.g., performing an action the robot was tasked with, or changing the articulated object's configuration), and the robot adapts to the new situation, in accordance with [43] and requirements R_3 and R_5.

3. Problem Statement and Reference Scenario

The problem we consider in this paper is three-fold: (i) given a target articulated object's configuration, determining a plan to attain such configuration as an ordered set of actions:

$$a = \{a_1, \ldots, a_i, \ldots, a_N; \prec\}, \tag{1}$$

where each action a_i involves one or more manipulation operations to be executed by a dual-arm robot, (ii) designing a planning and execution architecture for the manipulation of articulated objects, which is *efficient* and *flexible* in terms of perceptual features, their representation and action planning, and (iii) seamlessly integrating human actions *in the loop*, allowing the robot to adapt to novel, not planned beforehand, object's configurations on-line.

In order to provide planHRC with such a features, we pose a number of assumptions:

A_1 articulated objects (Figure 2) are characterised by an inertial behaviour, i.e., rotating one link causes the movement of all upstream or downstream links, depending on the rotation joint;

A_2 the effects of gravity on the articulated object's configurations are not considered, and the object is located on a table during all operations;

A_3 we do not assume any specific grasping or manipulation strategies to obtain a target object's configuration starting from another configuration; however, we do consider when an action a_i cannot be completed because of unexpected events or modelling omissions;

A_4 perception of articulated objects is affected by noise, but the *symbol grounding problem*, i.e., the association between perceptual features and the corresponding symbols in the robot's knowledge representation system [53], is assumed to be given.

As anticipated above, we need to represent articulated object's configurations. We define an articulated object as a 2-ple $\alpha = (\mathcal{L}, \mathcal{J})$, where $\mathcal{L}$ is the ordered set of its $|L|$ links, i.e., $\mathcal{L} = \{l_1, \ldots, l_j, \ldots, l_{|L|}; \prec\}$, and $\mathcal{J}$ is the ordered set of its $|J|$ joints, i.e., $\mathcal{J} = \{j_1, \ldots, j_j, \ldots, j_{|J|}; \prec\}$. Each link $l_j \in \mathcal{L}$ is characterized by two parameters, namely a length λ_l and an orientation θ_l. We allow only for a limited number of possible orientations. This induces an ordered set O of $|O|$ allowed orientation values, i.e., $O = \{o_1, \ldots, o_{|O|}; \prec\}$, such that an orientation θ_l can assume values in O. Given a link l_j, we define two sets, namely $up(l_j)$ and $down(l_j)$, such that the former is made of upstream links, i.e., from l_1 to l_{j-1}, whereas the latter includes downstream links from l_{j+1} to $l_{|J|}$.

Orientations can be expressed with respect to an *absolute*, possibly robot-centred reference frame, or – less intuitively – *relative* to each other, for instance θ_{l_i} can represent the rotation with respect to $\theta_{l_{i-1}}$. At a first glance, the absolute representation seems preferable because it leads to the direct perception of links and their orientations with respect to a robot-centred reference frame, whereas the set of absolute orientations constitute the overall object's configuration. When a sequence of manipulation actions are planned, changing one absolute orientation requires – in principle – the *propagation* of such change upstream or downstream the object via joint connections, which (hypothesis H_1) is expected to increase the computational burden on the reasoner and (H_2) may lead to suboptimal or redundant action sequences, because the propagation

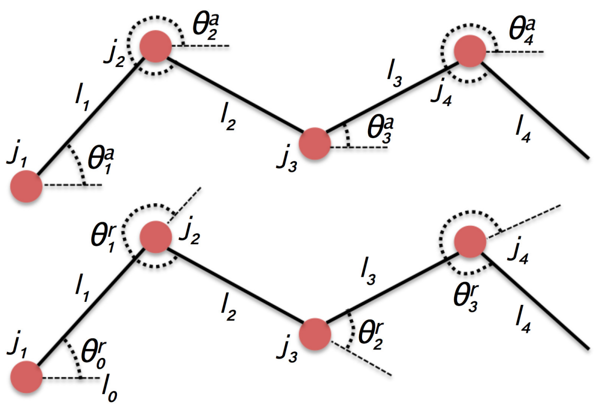

Figure 2: Two possible representations: absolute (top) and relative (bottom).

may jeopardise the effects of previous actions in the plan, or to sequences which cannot be fully understood by the human operator. On the contrary, the less intuitive *relative* approach assumes the direct perception of the relative orientations between pairwise links, and thus the overall object's configuration is made up of *incremental* rotations. In this case, (H_3) computation is expected to be less demanding, since there is no need to propagate one change in orientation to upstream or downstream links, and therefore (H_4) actions on different links *tend* to be planned sequentially. This has obvious advantages since it leads to shorter plans (on average), which could be further shortened by combining together action sub-sequences (e.g., two subsequent reorientations of 45 *deg* consolidated as one 90 *deg* single action), and to easy-to-understand plans.

If an articulated object is represented using absolute orientations (Figure 2 on the top), then its configuration is a $|L|$-ple:

$$\mathcal{C}_{\alpha,absolute} = \left(\theta_1^a, \ldots, \theta_l^a, \ldots, \theta_{|L|}^a\right), \tag{2}$$

where it is intended that the generic element θ_l^a is expressed with respect to an absolute reference frame. Otherwise, if relative angles are used (Figure 2 on the bottom), then the configuration must be *augmented* with an initial *virtual* link l_0 in order to define a reference frame, and therefore:

$$\mathcal{C}_{\alpha,relative} = \left(\theta_{0,virtual}^r, \theta_1^r, \ldots, \theta_l^r, \ldots, \theta_{|L|}^r\right). \tag{3}$$

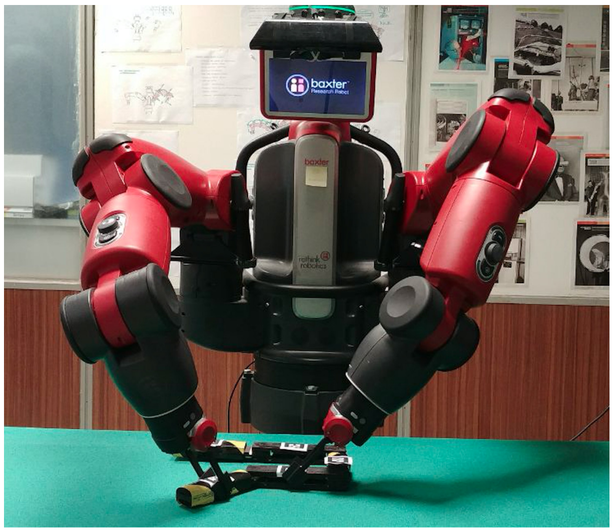

Figure 3: The experimental scenario: a Baxter dual-arm manipulator operating on an articulated object.

In principle, while the relative representation could model an object's configuration with one joint less compared to the absolute representation, the resulting configuration would not be unique (indeed there were infinitely many), since the object would maintain pairwise relative orientations between its links even when rotated *as a whole*. Therefore, an initial virtual reference link is added to the chain.

In order to comply with assumption A_2, we set up an experimental scenario where a Baxter dual-arm manipulator operates on articulated objects located on a table in front of it (Figure 3). Rotation operations occur only around axes centred on the object's joints and perpendicular to the table where the object is located. We have crafted a wooden articulated object made up of $|L| = 5$ 15.5 *cm* long links, connected by $|J| = 4$ joints. Links are 3 *cm* thick. The object can be easily manipulated by the Baxter's standard grippers, which complies with assumption A_3. To this aim, we adopt the MoveIt! framework. The robot is equipped with an RGB-D device located on top of its *head* pointing downward to the table. Only RGB information is used. QR tags are fixed to

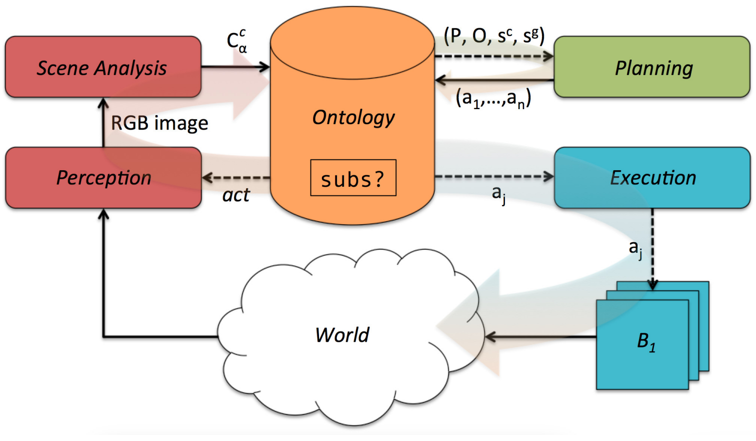

Figure 4: The information flow in planHRC.

each object's link, which is aimed at meeting assumption A_4. Each QR code provides a 6D link pose, which directly maps to an absolute link orientation θ^a_l. Finally, if relative orientations are employed, we compute them by performing an algebraic sum between the two absolute poses of two consequent links, e.g., $\theta^r_1 = |\theta^a_2 - \theta^a_1|$. A human can supervise robot operations and intervene when necessary from the other side of the table[2].

4. planHRC's Architecture

4.1. Information Flow

planHRC is organised as a number of parallel loops orchestrating the behaviour of different modules (Figure 4). Assuming that an articulated object α is located on the table in front of the robot, we want to modify its *current* configuration c^c_α to obtain a goal configuration c^g_α, which can be expressed using (2) or (3).

The goal configuration c^g_α is encoded as assertional knowledge in an OWL-based *Ontology* module [54]. When this happens, the *Perception* module is

[2]An example video is available at `https://www.youtube.com/watch?v=dMdzCB5FBMI`.

activated, and the Baxter's camera acquires an image of the workspace[3], which is fed to the *Scene Analysis* module. A *perceived* configuration c^p_α (i.e., the current configuration c^c_α) is extracted from the image, and a representation of it stored in the *Ontology* module. Both c^c_α and c^g_α are represented using conjunctions of class instances, which model such predicates as Connected, to indicate whether two links are connected by a Joint, or HasOrientation, to define angle orientations. If c^c_α and c^g_α *are different* then a planning process occurs. In order to determine such a difference, we assume the availability of a logic operator $\mathcal{D}$ that, given an element in the ontology, returns its description in OWL formalism. If the description of c^c_α is not *subsumed* by the description of c^g_α, i.e., it does not hold that $\mathcal{D}(c^c_\alpha) \sqsubseteq \mathcal{D}(c^g_\alpha)$, the *Planner* module is activated, which requires the definition of relevant predicates $\mathcal{P}_1, \ldots, \mathcal{P}_{|P|}$, and possible action types $\mathcal{A}_1, \ldots, \mathcal{A}_j, \ldots, \mathcal{A}_{|A|}$ in the form:

$$\mathcal{A}_j = \left(pre(\mathcal{A}_j), eff^-(\mathcal{A}_j), eff^+(\mathcal{A}_j)\right), \tag{4}$$

where $pre(\mathcal{A}_j)$ is the set of preconditions (in the form of predicates) for the action to be executable, $eff^-(\mathcal{A}_j)$ is the set of negative effects, i.e., predicates becoming false after action execution and $eff^+(\mathcal{A}_j)$ is the set of positive effects, i.e., predicates becoming true after execution. For certain domains, it is useful to extend (4) to allow for additional positive or negative effects, i.e., predicates becoming true or false in case certain additional conditions hold. A conditional action can be modelled as:

$$\mathcal{A}_j = \left(pre(\mathcal{A}_j), eff^-(\mathcal{A}_j), eff^+(\mathcal{A}_j), pre_a(\mathcal{A}_j), eff^-_a(\mathcal{A}_j), eff^+_a(\mathcal{A}_j)\right), \tag{5}$$

where $pre(\mathcal{A}_j)$, $eff^-(\mathcal{A}_j)$ and $eff^+(\mathcal{A}_j)$ are defined as before, $pre_a(\mathcal{A}_j)$ is the set of additional preconditions, whereas and $eff^-_a(\mathcal{A}_j)$ and $eff^+_a(\mathcal{A}_j)$ are the sets of additional effects subject to the validity of predicates in $pre_a(\mathcal{A}_j)$. Furthermore, the *Planner* requires a suitable description of the current state s^c

[3]The *Perception* module acquires images continuously, but for the sake of simplicity we treat each acquisition as if it were synchronous with action execution.

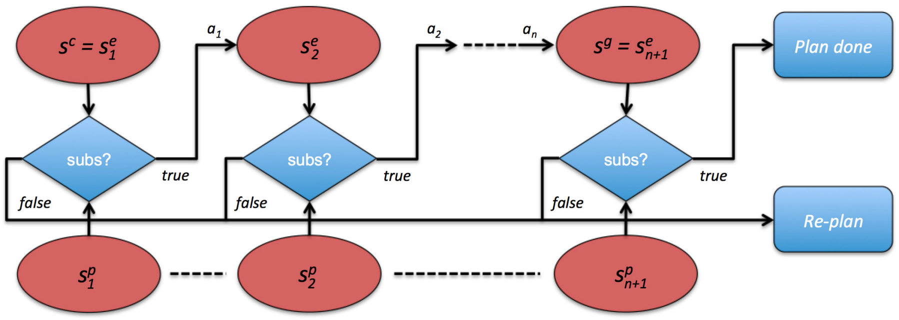

Figure 5: The planning and execution pipeline.

(including a description of c^c_α) and the goal state s^g (including c^g_α), described using an appropriate set of ground predicates $p_1, \ldots, p_{|p|}$. This information, encoded partly in the terminological section and partly in the assertional section of the *Ontology* module, is translated in a format the *Planner* module can use, namely the Planning Domain Definition Language (PDDL) [55].

A plan, as formally described in (1), is an ordered sequence of N actions whose execution changes the current state from s^c to s^g through a set of intermediate states. In a plan, each action corresponds to one or more scripted robot behaviours. For example, rotating a link l_{j+1} requires the robot to (i) keep the upstream link l_j steady with its left gripper, and (ii) rotate l_{j+1} of a certain amount with the right gripper. Such sequence shall not be encoded in the planning process, thereby reducing planning cost, but demanded to an action execution module. If a plan is found, each action is encoded in the ontology, along with all the *expected* intermediate states $s^c = s^e_1, s^e_2, \ldots, s^g = s^e_{N+1}$, which result from actions. The *Execution* module executes action by action activating the proper modules in the architecture, e.g., such *behaviours* as motion planning, motion execution, obstacle avoidance or grasping.

Each action a_j in a plan is assumed to transform a state s^e_j into a state s^e_{j+1}, such that:

$$s^e_{j+1} = \left(s^e_j \setminus eff^-(a_j)\right) \cup eff^+(a_j). \tag{6}$$

If a_j has additional conditions, then (6) is modified as:

$$s^e_{j+1} = \left(s^e_j \setminus \left(eff^-(a_j) \cup C^-(pre_a(a_j))\right)\right) \cup \left(eff^+(a_j) \cup C^+(pre_a(a_j))\right), \quad (7)$$

where conditions C^- and C^+ return the sets $eff^-_a(a_j)$ and $eff^+_a(a_j)$, respectively, if the conditions in $pre_a(a_j)$ hold, and $\emptyset$ otherwise. Before the action is executed, the *Ontology* module activates *Perception* to acquire a new image. Again, this induces a new perceived, current configuration c^c_α. Every time this happens, two situations can happen: if c^c_α corresponds to a current perceived state s^c whose description is subsumed by the description of a state s^e_{j-1} possibly generated applying an action a_{j-1} or as a consequence of human intervention, i.e., $\mathcal{D}(s^c) \sqsubseteq \mathcal{D}(s^e_{j-1})$, then the execution continues with action a_j until a state is reached which is subsumed by $\mathcal{D}(s^g)$; otherwise, a new planning process occurs, considering the current state s^c as a new initial state and keeping the previous goal state s^g.

A few remarks can be made. When an action a_j is executed, the expected intermediate state s^e_j is treated as a set of *normative* ground predicates, i.e., it defines the normal, expected state for a_j to be feasible. Whether s^e_j is obtained as a result of a previous action, or with the help of the human operator is not relevant for a_j. On the contrary, deviations from it are treated as violations and therefore the system tries to re-plan in order to reach a state compatible with s^g starting from the current state. As discussed above, violations can be of two kinds: on the one hand, human interventions (i.e., object manipulations on robot's behalf) may lead to a current state s^c not compatible with the expected intermediate state s^e_j, and therefore the robot should adapt by re-planning; on the other hand, a robot may not be able to complete action a_j, e.g., due to a cluttered workspace [17] or the obstructing presence of the human operator [7]. In the second case, if such an event were detected, the robot would re-plan starting from the current state, and possibly ask for the human operator's help to achieve a workable object's configuration. As a consequence, PLANHRC implements a policy according to which the overall system's performance is ensured by the use of state-of-the-art planning techniques, but it allows at any time the human operator to intervene and forces the robot to adapt its plan accordingly.

Figure 5 shows a graphical model of the information flow from the perspective of the planning process.

4.2. Reasoning in the Ontology, or the Cooperation Model

In planHRC, the *Ontology* module is used both *off-line* and *on-line* for different purposes[4]. The off-line use is related to modelling the domain of articulated objects manipulation, in terms of *types*, *predicates*, *operators*, *states*, *problems* and *plans*. The on-line use serves two purposes: on the one hand, to represent relevant object's configurations, such as the current c_α^c and the goal c_α^g configurations, as well as specific actions to perform using classes and relationships defined in the ontology; on the other hand, to apply such reasoning techniques as *instance checking* to the representation, e.g., to determine whether an action a_j assumes an expected planning state s_j^e which is compatible with the perceived current state s^c, as described in Figure 5.

As anticipated in Section 3, and in accordance with the findings in [41, 42, 43], the human-robot cooperation model implemented in planHRC foresees that: (i) the robot determines a plan maximising some performance indicator in terms of number of actions and/or time-to-completion; (ii) the robot executes and monitors each action in the plan; (iii) during normal work flow, the human operator supervises robot actions; and (iv) the human operator can intervene to cope with robot's failures in action planning or execution, or to perform tasks asynchronously and in parallel to robot activities. The model unfolding is based on monitoring the state transitions in (6) and (7) and their failures. Independently of the presence of conditional effects in an action a_j, two cases are possible after the action is submitted to the *Execution* module: it cannot be executed (or it is executed only in part) or it is carried out successfully.

[4]A more detailed description of the ontology is present in Appendix 1, whereas the full OWL ontology is available at [56].

The first case originates from motion planning or execution issues, e.g., because of a cluttered workspace [17] or to prevent any harm to the human operator [6, 7, 11]. If motion issues occur, planHRC does not generate a state compatible with s^e_{j+1}. However, this does not necessarily mean that the current state s^c is still compatible with the previous assessed state s^e_j, i.e., $\mathcal{D}(s^c) \sqsubseteq \mathcal{D}(s^e_j)$ may not hold, because the robot may have completed only part of the action. In this case, a new current state s^c is acquired. If there is an intermediate expected state s^e_i comparable with s^c, then execution resumes from action a_{i+1}; otherwise, it is necessary to invoke again the *Planner* module using $\mathsf{s^c}$ and $\mathsf{s^g}$, and obtain a new plan.

In the second case, action a_j is considered to be successful from the point of view of motion execution. Still, the outcome may or may not be compatible with the expected state s^e_{j+1}, e.g., due to not modelled effects. This state is observable as the current state s^c. However, although $\mathcal{D}(s^c) \sqsubseteq \mathcal{D}(s^e_{j+1})$ does not hold, it may happen that s^c could be appropriate for the next action a_{j+1} to occur. In particular, for a_{j+1} to be executable, it must hold that $\mathcal{D}(s^c) \sqsubseteq \mathcal{D}(pre(a_{j+1}))$. We treat the set of predicates in $pre(a_{j+1})$ as normative conditions for a_{j+1}, regardless whether the expected state s^e_{j+1} is generated as the outcome of the previous action a_j. If $\mathcal{D}(s^c) \sqsubseteq \mathcal{D}(pre(a_{j+1}))$ does not hold, we must check whether there is any intermediate expected state s^e_i comparable with s^c: if it is the case, execution resumes from action a_{i+1}; otherwise, re-planning is necessary.

In summary, human intervention is strictly necessary when a plan cannot be found. However, any human action is implicitly considered every time the current state does not comply with normative predicates.

4.3. Planning Models

As anticipated in Section 3, orientations can be expressed using an absolute or relative reference frame. These two possibilities lead to two planning models, which are characterized by different properties as far as (i) obtained plan, (ii) computational load of the planning process, and (iii) ease of execution for the robot, are concerned.

For the sake of description, we present the relative formulation first, and then the absolute one. The relative formulation employs the `:STRIPS` subset of PDDL, extended with `:equalities` and `:negative-preconditions`, whereas the absolute version requires also the use of `:conditional-effects`. Notably, the problem we are interested in induces a sort of granularity discretization of angular orientations, hence there is no practical necessity for continuous or hybrid planning models [57]. Therefore, PDDL constitutes an appropriate level of abstraction[5].

As discussed when introducing assumption A_1, our model assumes inertial behaviour, i.e., rotating one link affects the orientation of upstream or downstream links as well. Given a link l_j to rotate (clockwise or anticlockwise), two rotation actions are possible: (i) if link l_{j-1} is kept still and l_j is rotated (clockwise or anticlockwise), then all links in $down(l_j)$ rotate (clockwise or anticlockwise) and are displaced as well; (ii) if link l_{j+1} is kept still, all links in $up(l_j)$ are rotated (clockwise or anticlockwise) and displaced.

Each rotation action (either clockwise or anticlockwise) changing an angle θ_j^r referring to a relative orientation does not affect any other orientations of links in $up(l_j)$ or $down(l_j)$, since all of them are relative to each other, and therefore the planning process is computationally less demanding. However, since actions are expected to be based on link orientations grounded with respect to a robot-centred reference frame, i.e., absolute in terms of pairwise link orientations, a conversion must be performed, which may be greatly affected by perceptual noise, therefore leading to inaccurate or even inconsistent representations. In the absolute formulation, θ_j^a is considered absolute, and therefore it can be associated directly with robot actions. Unfortunately, this means that each action changing θ_j^a does affect numerically all other orientations of links in $up(l_j)$ or $down(l_j)$ in the representation, which must be kept track of using conditional effects in the planning domain.

[5]Examples of planning domains and problems can be found at [58].

```
(:action RotateClockwise
  :parameters (?l1 ?l2 - Link
    ?j1 - Joint ?o1 ?o2 - Orientation)
  :precondition (and
    (Connected ?j1 ?l1)
    (Connected ?j1 ?l2)
    (not (= ?l1 ?l2))
    (HasOrientation ?o1 ?j1)
    (OrientationOrd ?o1 ?o2))
  :effect (and
    (not (HasOrientation ?o1 ?j1))
    (HasOrientation ?o2 ?j1))
)
```

Figure 6: The *relative* version of RotateClockwise in PDDL.

Relative formulation. As described in Section 3, an articulated object α is represented using two ordered sets of links and joints. We use a `Connected` predicate modelled as described in (8) to describe the sequence of links in terms of binary relationships each one involving a link l_j and a joint j_{j+1}, which induces a pairwise connection between two links, namely l_j and l_{j+1}, since they share the same joint j_{j+1}. The orientation of a link l_j is associated with the corresponding joint j_j and corresponds to an angle θ^r_j, which ranges between 0 and 359 *deg*, using the predicate `HasOrientation` as specified in (9). This formulation assumes that link orientations are expressed incrementally relative to each other, and it implies that the robot's perception system is expected to provide the *Ontology* module with the set of relative link orientations as primitive information. If absolute link orientations are not available, the object's configuration $\mathcal{C}_{\alpha,absolute}$ can be computed applying forward kinematics formulas using relative orientations and link lengths. If noise affects the perception of link orientations, as it typically does, the reconstruction of the object's configuration may differ from the real one, and this worsens with link lengths. However, this model significantly simplifies the planning model's complexity: from a planner's perspective, the modification of any link orientations does not impact on other relative joint angles, and therefore rotation actions can be sequenced *in any order* the planner deems fit.

Angles are specified using *constants*, and are ordered using the predicate `OrientationOrd` as described by (10). The difference between constant values is the *granularity* of the resolution associated with modelled orientations. For example, if `30` and `45` are used as constants representing, respectively, a 30 and a 45 *deg* angle, then a predicate `(OrientationOrd 30 45)` is used to encode the fact that `30` precedes `45` in the orientation granularity, and corresponds to the description in (11).

Independently of what part of the articulated object is rotated, the domain model includes two actions, namely `RotateClockwise` (Figure 6) and `RotateAntiClockwise`. In the definition of `RotateClockwise`, `?l1` and `?l2` represent any two links l_j and l_{j+1}, `?j1` is the joint j_{j+1} connecting them, whereas `?o1` and `?o2` are the current and the obtained link orientations, respectively. If `?j1` connected two *different* links `?l1` and `?l2`, the angle `?o1` of `?l1` associated with `?j1` would be increased of a certain step (depending on the next orientation value) therefore leading to `?o2`. A similar description can be provided for `RotateAntiClockwise`.

A problem is defined by specifying the initial and final states. The former includes the *topology* of the articulated object in terms of `Connected` predicates, and its initial configuration using `HasOrientation` predicates; the latter describes its goal configuration using relevant `HasOrientation` predicates.

Absolute formulation. The absolute formulation differs from the relative one in that link orientations are expressed with respect to a unique, typically robot-centred, reference frame. If a rotation action modifies a given link orientation θ_j^a, all orientations of links in $up(l_j)$ or $down(l_j)$ must be consistently updated as well, i.e., it is necessary to propagate such change upstream or downstream. Such a representation increases the complexity of the planning task but it is more robust to errors: perceiving independent link orientations

```
(:action RotateClockwise
  :parameters (?l1 ?l2 - Link
    ?j1 - Joint ?o1 ?o2 - Orientation)
  :precondition (and
    (Connected ?j1 ?l1)
    (Connected ?j1 ?l2)
    (not (= ?l1 ?l2))
    (HasOrientation ?o1 ?j1)
    (OrientationOrd ?o1 ?o2))
  :effect
    (and
      (not (HasOrientation ?o1 ?j1))
      (OrientationOrd ?o2 ?j1)
      (forall (?j2 - Joint ?o3 ?o4 - Orientation)
        (when (and
           (Affected ?j2 ?l1 ?j1)
           (not (= ?j2 ?j1))
           (HasOrientation ?o3 ?j2)
           (OrientationOrd ?o3 ?o4))
        (and
           (not (HasOrientation ?o3 ?j2))
           (HasOrientation ?o4 ?j2)))
    )
)
```

Figure 7: The *conditional* version of `RotateClockwise` in PDDL.

induces an *upper bound* on the error associated with their inner angle. The `Connected`, `HasOrientation` and `OrientationOrd` predicates are the same as in the relative formulation, subject to the different semantics associated with link orientations. However, with respect to the relative formulation, the effects of the actions differ. In particular, the model assumes that we can represent which joints are affected when a link is rotated around one of the corresponding joints. This is done using the `Affected` predicate, i.e., a ternary predicate `(Affected ?j2 ?l1 ?j1)`, where `?l1` is the rotated link, `?j1` is the joint around which `?l1` rotates, and `?j2` is a joint affected by this rotation. Therefore, if `?j2` were affected, the angle of the corresponding link would be modified as well in the conditional statement and, as such, it would affect other joints via the corresponding links. For each couple `?l1`, `?j1`, the list of joints affected by the corresponding movement should be provided under the form of multiple `Affected` predicates. With reference to the action described in Figure 7, as in the previous case, the joint `?j1`, located between `?l1` and `?l2`, is increased by a quantity defined by a specific granularity, according to the `OrientationOrd` predicate. If rotating `?l2` around `?j1` affects `?j2`, the latter is updated, as well as all other joints upstream or downstream. This is encoded by the `forall` part of the PDDL encoding. Following the semantics of the language, the `forall` statement requires the planner to update the state of all joints `?j2` that are affected by the performed action – checked conditions are specified via the `when` statement. The `HasOrientation` predicate of identified affected joints is then updated accordingly. A similar definition for `RotateAntiClockwise` can be easily given.

In terms of problem definition, beside `Connected` and `HasOrientation` predicates, it is necessary to include the list of appropriately defined `Affected` predicates.

It is noteworthy that the two action definitions, namely `RotateClockwise` and `RotateAntiClockwise`, are functionally equivalent. Furthermore, any problem we target here could be solved – in principle – with just one action, as long as discretized angles were ring-connected. We decided to introduce two different actions for two reasons: on the one hand, it is rare that joints can rotate freely for 360 *deg* or more; on the other hand, this model leads to shorter plans (on average) in terms of number of actions and cleaner, more natural executions, at the expense of a slightly longer planning time.

5. Experimental Validation and Discussion

5.1. System Design

PLANHRC has been implemented integrating existing modules and novel *ad hoc* solutions. All experiments have been carried out using a dual-arm Baxter manipulator. The *Perception* and *Scene Analysis* modules are custom nodes developed using the Robot Operating System (ROS) framework. They integrates the Alvar tracker library to read QR codes [59]. Different solutions are equally legitimate, and the use of QR codes is not a fundamental features of the proposed framework. Images are collected using the standard RGB camera of a Kinect device, which is mounted on the Baxter's *head* and points downward to capture what happens on a table in front of the robot. The *Ontology* and *Planning* modules have been implemented on top of ROSPlan [39]. A custom ontology describing the domain of articulated object manipulation has been developed and validated. Ontology management is done using the ARMOR framework [60], which has been integrated with ROSPlan. Two existing planners have been interfaced with the system and evaluated, namely Probe [36] and Madagascar [37]. In principle, any existing PDDL-based planner with the features discussed above could be used. The two planners have been selected on the basis of their performance in the agile track of the 2014 International Planning Competition, as well as following a computational assessment of their performance with respect to other planners with similar features [61]. The *Execution* module and the various activated behaviours have been implemented using the well-known MoveIt! framework.

On-line, the architecture runs on a 8× Intel Core i7-4790 CPU 3.60 GHz processors workstation, with 8 GB of RAM, running a Linux Ubuntu 14.04.5 LTS operating system. Off-line performance tests about the planning process have been carried out on a workstation equipped with 2.5 GHz Intel Core 2 Quad processor, 4 GB of RAM, running a Linux 2.6.32 kernel operating system.

Problem formulations, as well as all generated instances, including domain, problems and plans, are freely available [58].

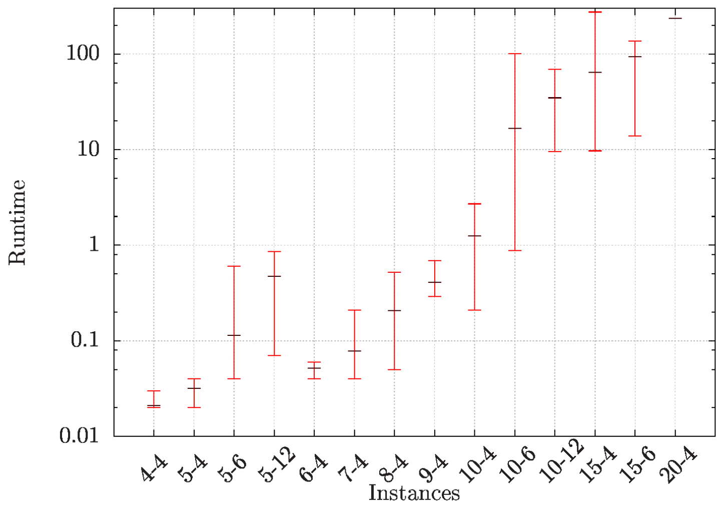

Figure 8: Means and variances of solution times for different problem instances using the absolute formulation and Probe: on the x-axis, the first value indicates the number of links, the second the number of allowed orientations. Runtime is reported in *seconds*.

5.2. Planning Performance

Tests with synthetic problem instances have been performed to stress the two planning formulations. For the tests, we varied the number of links $|L|$ from 4 to 20 and the number of allowed orientations $|O|$ a link can take from 4 (i.e., with a resolution of 90 *deg*) to 12 (i.e., with a resolution of 30 *deg*). As outlined above, such a resolution has a different meaning depending on whether we employ the absolute or relative formulations.

Figures 8 to 11 represent means and variances, in *seconds*, for different problem instances, for all the combinations of formulation and planner. Problem instances are labelled as $x - y$, where $x \leq |L|$ defines the number of links and $y \leq |O|$ specifies the orientation resolution. For each instance, planners have been executed 10 times to take into account the randomness associated with the employed heuristics. A 300 *sec* upper bound to the solution time has been set. If a planner is unable to find a solution before such time limit is reached, it is stopped. Figures only contain data related to problems solved within the time

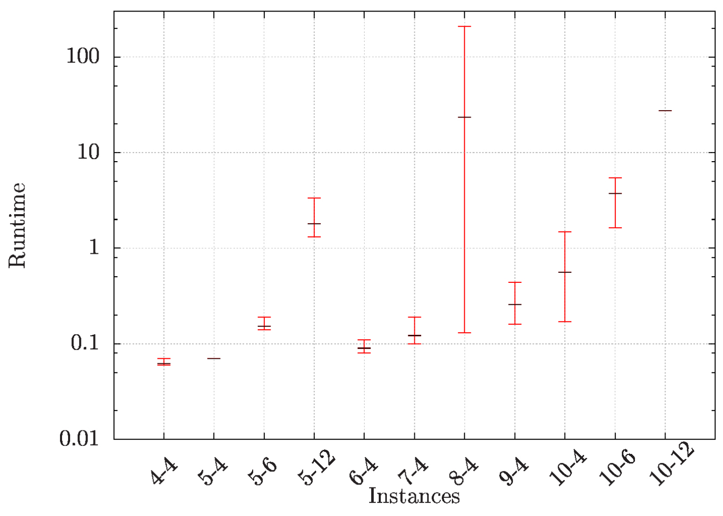

Figure 9: Means and variances of solution times for different problem instances using the absolute formulation and Madagascar: on the x-axis, the first value indicates the number of links, the second the number of allowed orientations. Runtime is reported in *seconds*.

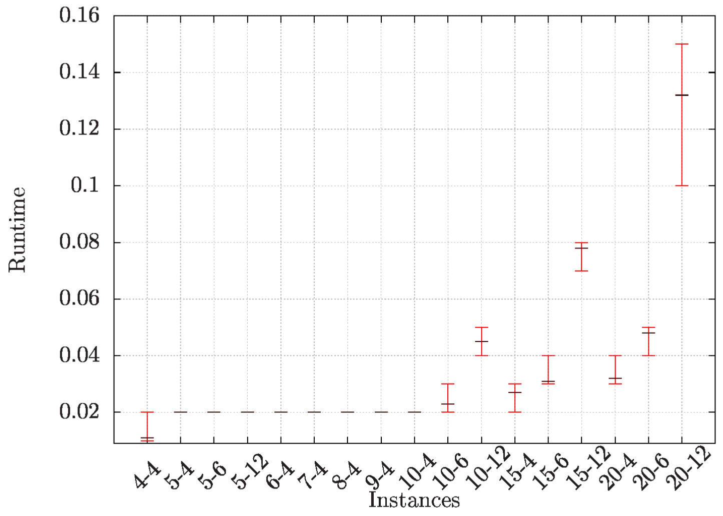

Figure 10: Means and variances of solution times for different problem instances using the relative formulation and Probe: on the x-axis, the first value indicates the number of links, the second the number of allowed orientations. Runtime is reported in *seconds*.

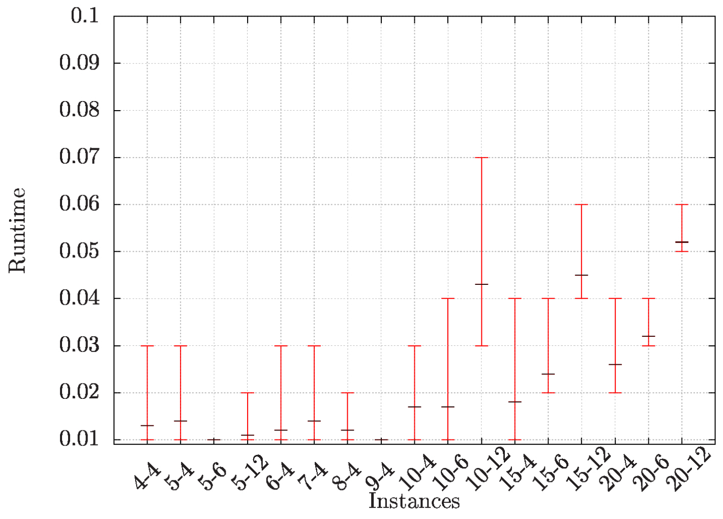

Figure 11: Means and variances of solution times for different problem instances using the relative formulation and Madagascar: on the x-axis, the first value indicates the number of links, the second the number of allowed orientations. Runtime is reported in *seconds*.

limit.

As it can be seen in Figure 8, when we use the absolute formulation and Probe, 73.5% of the instances are solved, i.e., 125 out of 170. It is possible to observe that problem instances with up to $x \leq 10$ and $y \leq 4$ are solved in roughly less than 1 *sec*, with a relatively small variance. When the number of links increase, planning time significantly increases as well, and thus the variance. In the same situation, as depicted in Figure 9, Madagascar shows a more unpredictable behaviour: for small problem instances, it can quickly find a solution, and with a small temporal variance; however, the employed heuristics may cause large variances in specific cases, e.g., the instance labelled $8 - 4$. It is worthy to note that larger instances are rarely solved and, in general, the number of solved instances is lower when compared to Probe, i.e., only 53.5% (91 out of 170). As it will be also showed in the next Section, these results seem to confirm hypothesis H_1, i.e., the more intuitive absolute formulation leads to more complex reasoning processes. This is due to the fact that planners need to propagate the effects of each action to upstream or downstream links, which can be done only by employing a complex formulation involving conditional effects.

If we consider the relative formulation, approach, then both Probe (Figure 10) and Madagascar (Figure 11) are very efficient, with Madagascar outperforming Probe to a small extent. Both planners are capable of solving all the instances (170 out of 170) in less that 0.2 *sec*, and exhibit a very good scalability, as well as a very limited variance. These results support hypothesis H_3, i.e., the reduced planning effort is reflected by the simpler formulation.

5.3. Examples

In this Section, we provide examples of plans generated by Probe and Madagascar using the two formulations introduced above. Furthermore, we show and discuss what happens in a number of human-robot cooperation use cases.

In order to discuss how the different planners deal with the absolute and the relative formulation, we focus the discussion on a specific instance with 3 links and 3 joints. Figure 12 shows two possible solutions, obtained respectively using Probe (first two rows) and Madagascar (last two rows), when the absolute formulation is adopted. In each solution, the top-leftmost configuration is the initial one, whereas the bottom-rightmost configuration is final one. It can be observed that both plans are characterized by a number of seemingly unnecessary actions, since the planners must continuously maintain the representation consistency. The plan obtained using Madagascar (on the bottom) also loops over two configurations, which is probably due to the employed heuristics. This example seems to confirm H_2, i.e., the absolute approach leads to suboptimal plans, or plans which may not easily understood by human co-workers.

Figure 13 shows how Probe (top) and Madagascar (bottom) solve the same problem when a relative formulation is adopted. Both planners generate solutions that are shorter than those obtained using the the absolute formulation, and no seemingly unnecessary actions are planned. In the plan generated by Madagascar, it is possible to observe that actions involving the same link tend to be performed sequentially, i.e., H_4 seems to be verified. This holds for other

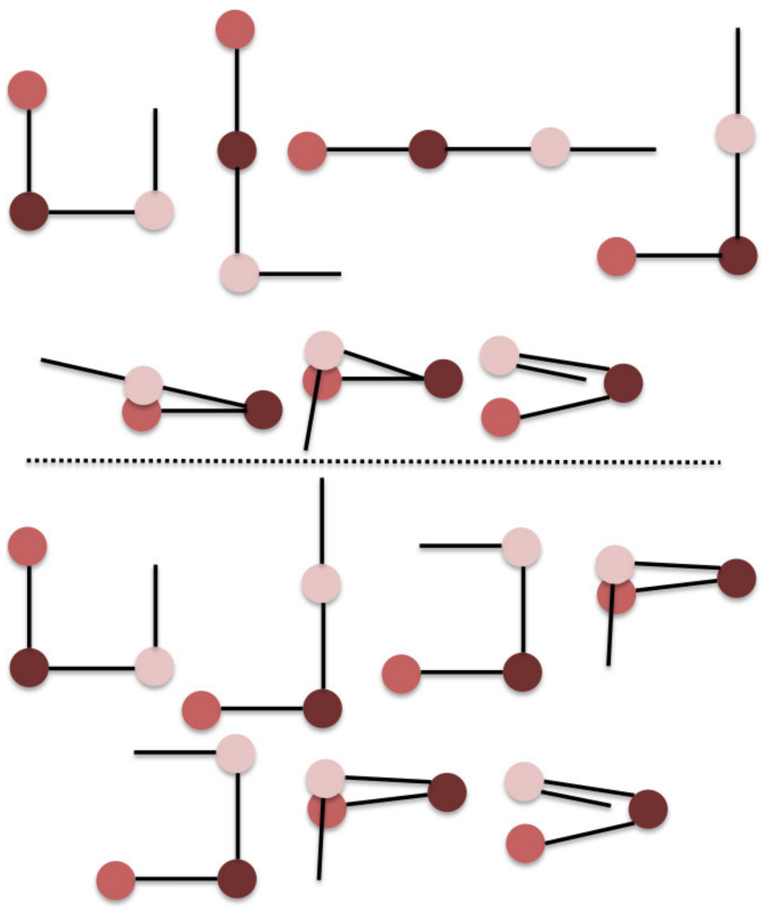

Figure 12: A sequence of configurations for a $3-3$ problem, using the absolute formulation with Probe (first two rows, from left to right and top to bottom) and Madagascar (second two rows, from left to right and top to bottom).

solutions as well.

As anticipated above, PLANHRC has been deployed on a dual-arm Baxter manipulator to enable the robot to autonomously manipulate articulated objects. The Baxter operates on a 3-link articulated object, assuming that the angle resolution is 90 *deg*, i.e., a $3-4$ problem according to the definition introduced above. Figure 14 shows a sequence of configurations, including the initial one in the top-leftmost position, and the goal one in the bottom-rightmost position, from left to right and top to bottom, whereas Figure 15 shows the cor-

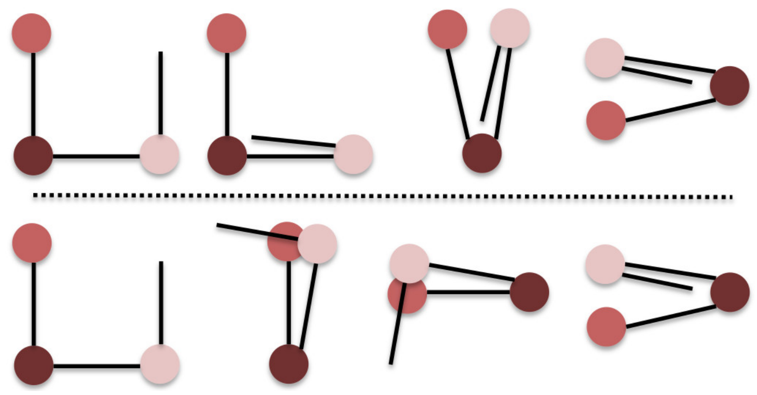

Figure 13: A sequence of configurations for a $3-3$ problem, using the relative formulation with Probe (first row, from left to right) and Madagascar (second row, from left to right).

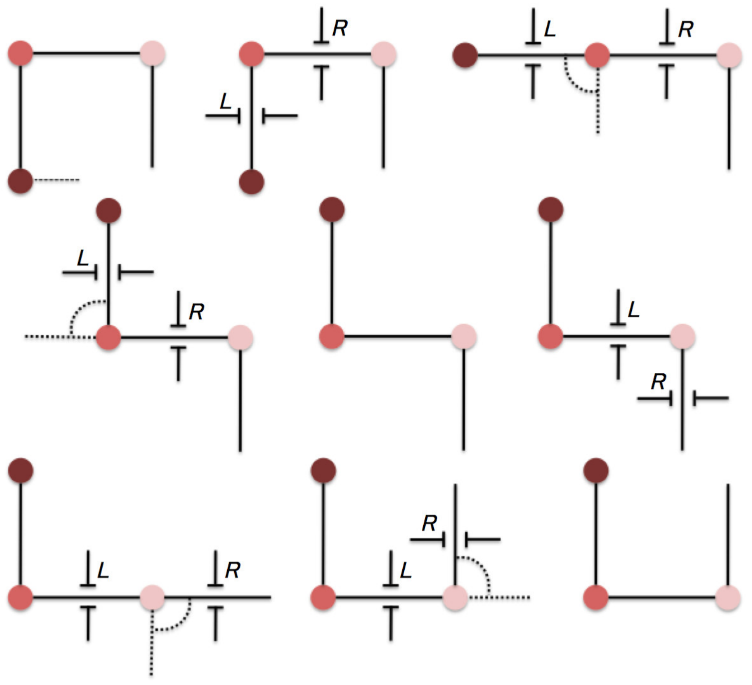

Figure 14: A sequence of configurations for a $3-4$ problem, from left to right and top to bottom, as seen from the robot's perspective.

responding relevant instants during the execution of the plan by the robot. It is worth noting that, each time a RotateClockwise or RotateAntiClockwise action

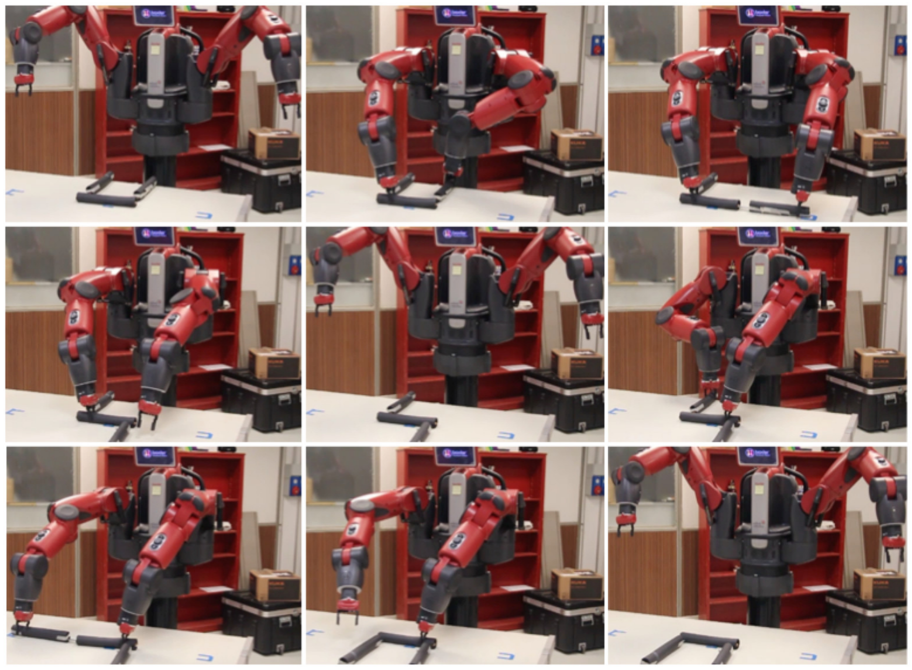

Figure 15: The sequence of Figure 14 as executed by Baxter without human intervention.

is executed, the actual robot behaviour is made up of three steps: the first is to firmly *grasp* the link associated with the interested joint that must be kept still, the second is to *grasp* the link that must be rotated, the third is the actual *rotation* of the proper amount. In planHRC, this can be done indifferently by the left or right robot arms, according to a simple heuristics related to which arm is closer to the link to operate on. Grasping actions in Figure 14 are indicated with grasping signs close to the interested link, plus an R sign to indicate that the action is performed with the right arm, or L otherwise. We decided not to model grasping actions at the planning level for two reasons: on the one hand, they would have increased the burden of the planning process; on the other hand, each rotation must be preceded by a grasping operation, and therefore this sequence can be easily serialized in the execution phase.

Figure 16 and Figure 17 show two examples of plans where human intervention occurs to successfully accomplish the whole cooperation process. In the

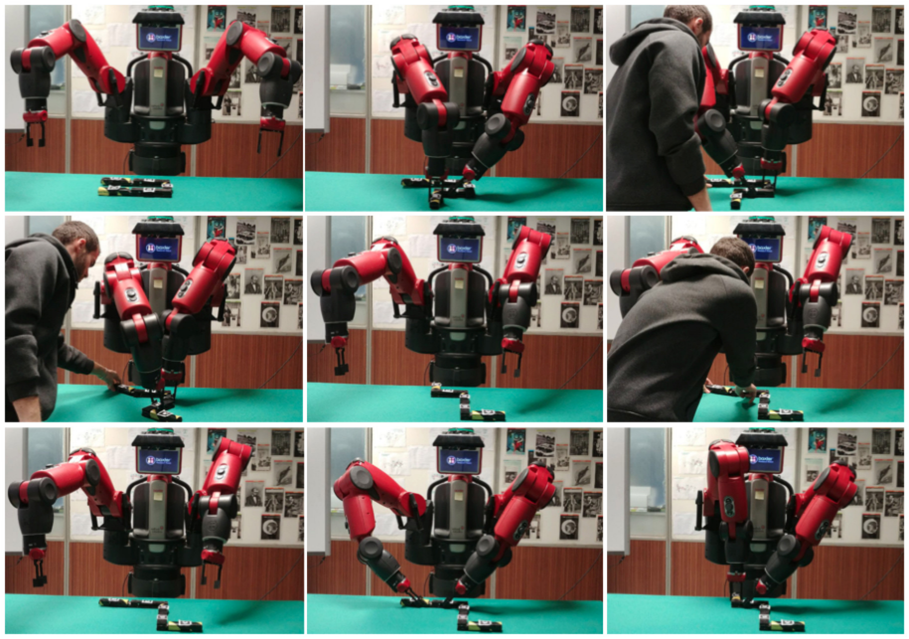

Figure 16: A series of manipulation actions executed with the help of a human operator.

figures, the two sequences must be analysed from top to bottom and left to right.

In Figure 16, it is possible to see that the human operator performs an action *while* the robot is executing a rotation action on other links (top-right and mid-left snapshot). The action performed by the human operator leads to a situation compatible with the object's target configuration. As a consequence, the final configuration is reached in snapshot mid-centre. Afterwards, the operator modifies again the status of the first link (mid-right snapshot), thereby leading to a configuration not compatible with the goal one. As a consequence, the robot intervenes to restore it (bottom-centre and bottom-right snapshots). This sequence demonstrates two important features of planHRC: first, the freedom human operators have in performing actions asynchronously with robot actions; second, the robot capabilities in keeping the cooperation *on track* coping with possible human mistakes.

Figure 17 shows an example where a human operator helps the robot com-

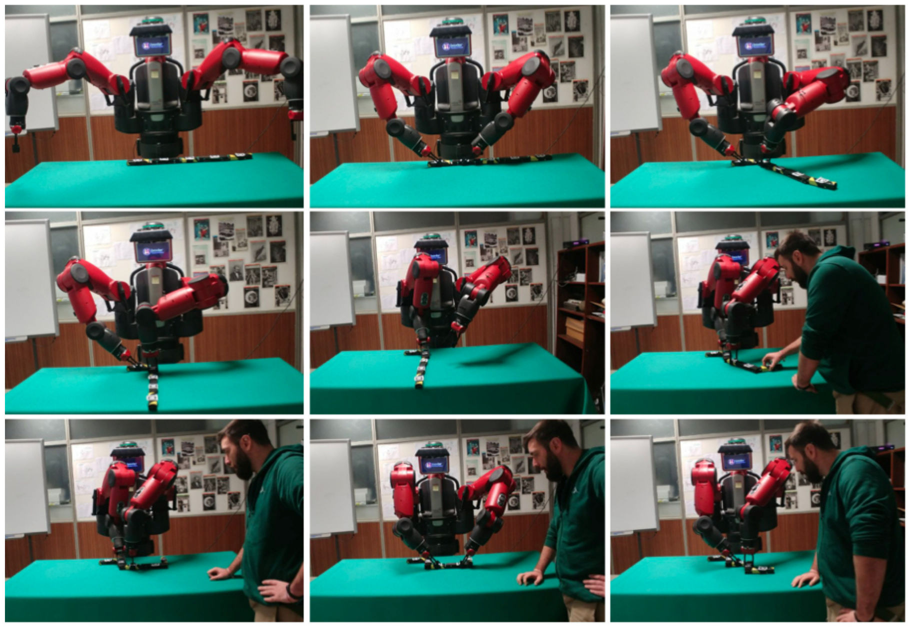

Figure 17: Another series of manipulation actions executed with the help of a human operator.

plete an action, which was not performed in its entirety. The robot starts executing a plan (top-left and top-centre snapshots). However, a rotation action is not completed, leading the object's configuration to a state not compatible with the expected one (mid-right snapshot). Then, the human operator intervenes with an action aimed at completing the intended rotation and, at the same time, performing an additional rotation on the last link in the chain (mid-right snapshot). From that moment on, the robot autonomously completes the plan. This sequence shows how a plan can be successfully recovered by human intervention, and the fact that the robot can seamlessly continue plan execution.

5.4. Discussion and Comparison with Other Approaches

On the basis of the requirements outlined above and the experimental analysis carried out to evaluate the whole planHRC architecture, it is possible to make a few interesting remarks, perform a comparison with other approaches in the literature, and draw some conclusions. In particular, the discussion that follows is focused on three aspects, namely planning performance, the generation of *natural* sequences of manipulation actions and the resulting cooperation process according to which human operators interact with the robot.

Planning performance. The absolute and the relative formulations are characterized by different performance results.

When using the absolute formulation, both Probe and Madagascar are capable of solving problem instances with a limited number of links and orientations in less than 1 *secs*, which is a reasonable upper bound for the reasoning time of a collaborative robot interacting with a human operator, with Probe outperforming Madagascar on bigger problem instances. With around 10 links, the time required to obtain a plan (if it exists) significantly increases, due to the large number of possible orientations, with solution times up to an average of 100 *secs* and beyond. When using Probe, solution times for the same problem instance have a certain variance, which is almost uniform for different numbers of links and possible orientations. If Madagascar is used, such variance generally decreases, but sometimes it may become significantly large, as shown for example in the problem instance $8-4$. By carefully analysing cases where Madagascar shows significantly high runtimes, we observed that the planner finds problem instances where subsequent connected joints need to be rotated in opposite ways (e.g., the angle of one joint has to be decreased, while the angle of the other joint has to be increased) particularly challenging to solve. In that cases, the planner keeps looping between a very small number of configurations, trying to fix the orientation of a joint at a time, ignoring the effect of such actions on the rest of the articulated object. As far as human-robot cooperation processes are concerned, if an absolute formalization were used, then Probe would represent the best trade-off between complexity and solution times. In principle, Madagascar would be a better choice for problems with a reduced number of links and possible orientations, but the occasional presence of large variances in solution times would seriously jeopardize the human-robot cooperation process. The two planners behave differently when using a relative formulation. Both Probe and Madagascar prove capable of solving large problem instances (i.e., with up to 20 links and up to 12 possible orientations) in less than 0.2 *secs*. Solution times are exponential also in this case, but the very low time scale makes such trend relevant only to a limited extent. Differently from the case with the absolute formulation, Probe behaves quite deterministically, and the same holds for Madagascar. When dealing with human-robot cooperation, both planners are suitable to be used if a relative formulation is adopted, with a slight preference for Probe.

The relative formulation proves to be essential when the robot must deal with the directive D_2 discussed in the Introduction, and in particular to allow for a fast action re-planning when needed, as required by R_3.

Differently from those approaches encoding human operator preferences in the planning model, typically using heuristics [43], when using a relative formulation planHRC tends to find minimum-length plans (in terms of number of actions), i.e., the plan as devised by the robot is *efficient*. Human operator preferences are then taken into account on-line. As a matter of fact, interventions of human operators are treated by planHRC as *perturbations* with respect to the execution of the efficient plan. However, sometimes these perturbations may be helpful (i.e., the human operator helps the robot perform an action), whereas in other cases they constitute *detours* with respect to the original plan, which is tolerated because such detours express human operator preferences. Differently from the approaches presented in [15, 44, 45], planHRC does not model human preferences in the planning models, but accommodates for them on-line. Only to a limited extent, the approach presented in [11] goes in the direction pursued by planHRC. The use of AND/OR graphs to model a limited number of alternative cooperation models allows human operators to select on the fly which one they want to follow. However, the AND/OR graph encodes models which have been *a priori* defined, and this is different from the approach of planHRC where (i) there is no need for such an encoding, and (ii) in principle, the cooperation is not limited to a given number of alternatives.

Natural action sequences. In general, the two formulations lead to qualitatively different plans, i.e., plans with different actions.

Independently of the employed planner, the absolute formulation originates plans longer than those obtained using the relative formulation. In the absolute case, the solution may contain apparently unnecessary actions, as well as repeated sequences of actions. This is due to the fact that when working on orientations of links located downstream in the chain, such orientations may be later modified as a side-effect when the algorithm operates on links upstream, therefore requiring reworking on downstream links. Such plans are the result of certain planner heuristics. However, they are often unnatural for humans to understand, which is of the utmost importance in human-robot cooperation processes.

Plans obtained starting from the relative formulation are shorter and – in a generic sense – more understandable by humans. Since the representation of orientations is relative for pairwise links, the planner does not need to modify orientations of downstream links multiple times, and solutions tend to include sequences of actions operating on the same link. This makes plans easy to follow, irrespectively whether they are generated using Probe or Madagascar.

Thus, as far as naturalness is concerned, the relative formulation must be preferred over the absolute formulation. Shorter and easy-to-understand plans are supposed to strengthen a human operator's ability to supervise robot actions in compliance with directive D_2 and to intervene when required, as prescribed by requirement R_5. However, it is noteworthy that planHRC has not been tested in real-world conditions yet. As a consequence, there are still to-be-validated hypotheses requiring us to conduct a specifically designed study, also related to the role of context-aware planning in human-robot cooperation [62].

According to the studies discussed in [41, 42], human operators tend to prefer a partial control on the cooperation process, with the aim of maximising the overall human-robot team's performance. The approach pursued by planHRC goes in this direction in that it enables the robot to generate an efficient plan, but it allows humans to intervene when required. If compared to those approaches explicitly or implicitly encoding human preferences in the cooperation process [10, 11, 15, 44, 45], planHRC does not offer any formal guarantee about the naturalness of the generated plan, that is to say in terms of an easy understanding of the sequence of basic manipulation actions by human operators. However, when a relative formulation is adopted, the planner tends to produce natural, easy-to-understand plans without *prior* knowledge being encoded in the system, which is a clear advantage should the system be extended to other use cases.

The cooperation process. In absence of errors related to action execution, once a plan is available planHRC should be able to carry it out in its entirety. This is in agreement with directive D_1 discussed in the Introduction. However, when either one action is not executed successfully or it has been carried out only partially, a human operator can intervene to obtain an object configuration that the robot can operate upon. These two facts support requirement R_2.

As described above, before any action is executed, the robot checks whether a number of expected normative predicates hold in the current planning state. Implicitly, this means that any error in action execution or human intervention is synchronously assessed before the next planned action can start. Obviously enough, this represents a limiting factor for planHRC, and originates from the focus on planning sequences of *states* to be reached rather than actions. A more flexible reactive system may make use of human actions to determine causes of faults on the fly, instead of being limited in assessing their outcomes at discrete intervals. However, it also enforces the fact that humans are in control at any time: the robot simply waits for human intervention to finish and then plans a course of action from that moment on.

This approach makes planHRC different from a number of human-robot cooperation frameworks described in the literature [47, 48, 49, 50, 51, 52]. While in [47] a predefined set of possible cause-effect events are considered, planHRC consider each predicate in the ontology as normative information that must be validated on-line, independently of the cause that may have generated a norm violation. planHRC does not explicitly detect human operator actions [48, 50, 51], and therefore it is not able to perform action-dependent behaviour, but only state-dependent behaviour. In virtue of this, planHRC may be employed to perform anticipative behaviours like done in [50], where a Bayesian network is employed to that aim, but using only the current cooperation state (i.e., adopting a sort of Markov assumption). It is noteworthy that planHRC explicitly does not consider temporal aspects in planning execution. Whilst – in principle – temporal PDDL-based planners may be used to generate plans adhering with well defined temporal constraints, at run time planHRC may support the use of temporal-based constraints validation as done, for instance, in [49].

Also in this case, the benefits of this approach should be validated with human operators. Current work is devoted to investigate these matters.

6. Conclusions

The paper proposes a hybrid reactive/deliberative architecture for collaborative robots in industrial scenarios, and it shows a use case where a human and a robot collaboratively manipulate articulated objects.

The paper contributes to the literature in two respects: (i) it shows how two different representation and planning models for articulated objects impact on planning performance and plan quality, in terms of number of actions and simplicity of the plan; (ii) it demonstrates the feasibility of an approach to human-robot cooperation where actions by human operators are automatically managed in virtue of their effects as perceived by the robot.

The developed architecture is evaluated on the basis of a number of functional and non functional requirements: the possibility for the system to implicitly recognise the effects of human actions, the robot's capabilities in adapting to those actions, and a fast (re-)planning process when needed, just to name the most important ones.

Current work is planned to address three aspects: the first is related to a more detailed, computationally efficient, representation of articulated objects and the corresponding planning models. The second focuses on the investigation of planning models represented using more expressive languages than

PDDL, such as the PDDL+ language [57]. Finally, the third requires a systematic evaluation of the employed human-robot cooperation process with human volunteers.

References

[1] J. Krüger, T. Lien, A. Verl, Cooperation of humans and machines in the assembly lines, CIRP Annals - Manufacturing Technology 58 (2) (2009) 628–646.

[2] C. Heyer, Human-robot interaction and future industrial robotics applications, in: Proceedings of the 2010 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan, 2010.

[3] P. Tsarouchi, A. Matthaiakis, S. Makris, G. Chryssolouris, Human-robot interaction review and challenges on task planning and programming, International Journal of Computer Integrated Manufacturing 29 (8) (2016) 916–931.

[4] J. Baraglia, M. Cakmak, Y. Nagai, R. Rao, M. Asada, Initiative in robot assistance during collaborative task execution, in: Proceedings of the 2016 ACM/IEEE International Conference on Human-Robot Interaction (HRI 2016), Christchurch, New Zealand, 2016.

[5] T. Munzer, Y. Mollard, M. Lopes, Impact of robot initiative on human-robot collaboration, in: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction (HRI 2017), Vienna, Austria, 2017.

[6] S. Denei, F. Mastrogiovanni, G. Cannata, Towards the creation of tactile maps for robots and their use in robot contact motion control, Robotics and Autonomous Systems 63 (3) (2015) 293–308.

[7] S. Haddadin, E. Croft, Physical human-robot interaction, in: B. Siciliano and O. Khatib (Eds.) Springer Handbook of Robotics, Springer International Publishing, 2016.

[8] T. Chakraborti, S. Kambhampati, M. Scheutz, Y. Zhang, AI challenges in human-robot cognitive teaming, arXiv preprint arXiv:1707.04775.

[9] M. Goodrich, A. Schultz, Human-robot interaction: a survey, Foundations and Trends in Human-Computer Interaction 1 (3) (2007) 203–275.

[10] L. Johannsmeier, S. Haddadin, A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes, IEEE Robotics and Automation Letters 2 (1) (2017) 21–48.

[11] K. Darvish, F. Wanderlingh, B. Bruno, E. Simetti, F. Mastrogiovanni, G. Casalino, Flexible human-robot cooperation models for assisted shop-floor tasks, arXiv preprint arXiv:1707.02591.

[12] K. Dautenhahn, Socially intelligent robots: dimensions of human-robot interaction, Philosophical Transactions of the Royal Society of London, Series B, Biological Sciences 362 (1480) (2007) 679–704.

[13] M. Prewett, R. Johnson, K. Saboe, L. Elliott, M. Coovert, Managing workload in human-robot interaction: a review of empirical studies, Computers in Human Behaviour 26 (5) (2010) 840–856.

[14] A. Clair, M. Matarić, How robot verbal feedback can improve team performance in human-robot as collaborations, in: Proceedings of the 2015 ACM/IEEE International Conference on Human-Robot Interaction (HRI 2015), Portland, USA, 2015.

[15] A. Roncone, O. Mangin, B. Scassellati, Transparent role assignment and task allocation in human robot collaboration, in: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA 2017), Singapore, 2017.

[16] H. Liu, L. Wang, Gesture recognition for human-robot collaboration: a review, International Journal of Industrial Ergonomics - in press.

[17] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, P. Abbeel, Combined task and motion planning through an extensible planner-independent interface layer, in: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014), Hong Kong, China, 2014.

[18] D. Henrich, H. Worn, Robot manipulation of deformable objects, Advanced Manufacturing, Springer-Verlag, London, Berlin, Heidelberg, 2000.

[19] M. Saadat, P. Nan, Industrial applications of automatic manipulation of flexible materials, Industrial Robot: an International Journal 29 (5) (2002) 434–442.

[20] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. Dimarogonas, D. Kragic, Dual arm manipulation: a survey, Robotics and Autonomous Systems 60 (10) (2012) 1340–1353.

[21] P. Jimenez, Survey on model-based manipulation planning of deformable objects, Robotics and Computer-Integrated Manufacturing 28 (2) (2012) 154–163.

[22] H. Wakamatsu, E. Arai, S. Hirai, Knotting and unknotting manipulation of deformable linear objects, International Journal of Robotic Research 25 (4) (2006) 371–395.

[23] A. Nair, D. Chen, P. Agraval, P. Isola, P. Abbeel, J. Malik, S. Levine, Combining self-supervised learning and imitation for vision-based rope manipulation, in: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA 2017), Singapore, 2017.

[24] Y. Yamakawa, A. Namiki, M. Ishikawa, Dynamic high-speed knotting of a rope by a manipulator, International Journal of Advanced Robotic Systems 10 (2013) 1–12.

[25] L. Bodenhagen, A. Fugl, A. Jordt, M. Willatzen, K. Andersen, M. Olsen, R. Koch, H. Petersen, N. Kruger, An adaptable robot vision system performing manipulation actions with flexible objects, IEEE Transactions on Automation Science and Engineering 11 (3) (2014) 749–765.

[26] J. Schulman, J. Ho, C. Lee, P. Abbeel, Learning from demonstrations through the use of non-rigid registration, in: M. Inaba and P. Corke (Eds.) Robotics Research, Vol. 114 of Springer Tracts in Advanced Robotics, Springer International Publishing, Lausanne, Switzerland, 2016.

[27] S. Miller, J. van den Berg, M. Fritz, T. Darrell, K. Goldberg, P. Abbeel, A geometric approach to robotic laundry folding, International Journal of Robotic Research 31 (2) (2011) 249–267.

[28] D. Berenson, Manipulation of deformable objects without modelling and simulating deformation, in: Proceedings of the 2013 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2013), Tokyo, Japan, 2013.

[29] S. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, D. Caldwell, Learning symbolic representations of actions from human demonstrations, in: Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA 2015), Seattle, WA, USA, 2015.

[30] A. Howard, G. Bekey, Recursive learning for deformable object manipulation, in: Proceedings of the 1997 International Conference on Advanced Robotics (ICAR 1997), Monterey, CA, USA, 1997.

[31] F. Mastrogiovanni, A. Sgorbissa, R. Zaccaria, A system for hierarchical planning in service mobile robotics, in: Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8), Amsterdam, The Netherlands, 2004.

[32] B. Frank, R. Schmedding, C. Stachniss, M. Teschner, W. Burgard, Learning the elasticity parameters of deformable objects with a manipulation

robot, in: Proceedings of the 2010 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Taipei, Taiwan, 2010.

[33] C. Elbrechter, R. Haschke, H. Ritter, Bi-manual robotic paper manipulation based on real-time marker tracking and physical modelling, in: Proceedings of the 2011 IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS 2012), San Francisco, CA, USA, 2011.

[34] C. Elbrechter, R. Haschke, H. Ritter, Folding paper with anthropomorphic robot hands using real-time physics-based modeling, in: Proceedings of the 2012 IEEE-RAS International Conference on Humanoid Robotics (HUMANOIDS 2012), Osaka, Japan, 2012.

[35] M. Fox, D. Long, PDDL 2.1: an extension to PDDL for expressing temporal planning domains, Journal of Artificial Intelligence Research 20 (2003) 61–124.

[36] N. Lipovetzky, H. Geffner, Searching for plans with carefully designed probes, in: Proceedings of the 2011 International Conference on Automated Planning and Scheduling (ICAPS 2011), Freiburg, Germany, 2011.

[37] J. Rintanen, Madagascar: scalable planning with SAT, in: Proceedings of the 2014 International Planning Competition (IPC 2014), Portsmouth, NH, USA, 2014.

[38] M. Fox, R. Howey, D. Long, Validating plans in the context of processes and exogenous events, in: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005), Pittsburgh, Pennsylvania, USA, 2005.

[39] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, M. Carreras, ROSPlan: planning in the Robot Operating System, in: Proceedings of the 2015 International Conference on Automated Planning and Scheduling (ICAPS 2015), Jerusalem, Israel, 2015.

[40] I. Susan, S. Chitta, MoveIt!, `http://moveit.ros.org`.

[41] M. Gombolay, R. Wilcox, J. Shah, Fast scheduling of multi-robot teams with temporospatial constraints, in: Proceedings of Robotics: Science and Systems IX (RSS 2013), Berlin, Germany, 2013.

[42] M. Gombolay, R. Gutierrez, G. Starla, J. Shah, Decision making, authority, team efficiency and human worker satisfaction in mixed human-robot teams, in: Proceedings of Robotics: Science and Systems X (RSS 2014), Berkeley, USA, 2014.

[43] M. Gombolay, C. Huang, J. Shah, Coordination of human-robot teaming with human task preferences, in: Proceedings of the 2015 AAAI Fall Symposium Series, Palo Alto, CA, USA, 2015.

[44] M. Cirillo, L. Karlsson, A. Saffiotti, Human-aware task planning, ACM Transactions on Intelligent Systems and Technology 1 (2) (2010) 1–26.

[45] R. Wilcox, S. Nikolaidis, J. Shah, Optimisation of temporal dynamics for adaptive human-robot interaction in assembly manufacturing, in: Proceedings of Robotics: Science and Systems VIII (RSS 2012), Sydney, Australia, 2012.

[46] E. Sebastiani, R. Lallement, R. Alami, L. Iocchi, Dealing with on-line human-robot negotiations in hierarchical agent-based task planner, in: Proceedings of the 2017 International Conference on Automated Planning and Scheduling (ICAPS 2017), Pittsburgh, USA, 2017.

[47] A. Agostini, C. Torras, F. Wörgötter, Integrating task planning and interactive learning for robots to work in human environments, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11), Barcellona, Spain, 2011.

[48] H. Koppula, A. Jain, A. Saxena, Anticipatory planning for human-robot teams, in: Proceedings of Robotics: Science and Systems IX (RSS 2013), Berlin, Germany, 2013.

[49] E. Karpas, S. Levine, P. Yu, B. Williams, Robust execution of plans for human-robot teams, in: Proceedings of the 2015 International Conference on Automated Planning and Scheduling (ICAPS 2015), Jerusalem, Israel, 2015.

[50] C. Liu, J. Fisac, Goal inference improves objective and perceived performance in human-robot collaboration, in: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey, 2015.

[51] W. Kwon, I. Suh, Planning of proactive behaviours for human-robot cooperative tasks under uncertainty, Knowledge-based Systems 72 (2014) 81–95.

[52] R. Caccavale, A. Finzi, Flexible task execution and attentional regulations in human-robot interaction, IEEE Transactions on Cognitive and Developmental Systems 9 (1) (2017) 68–79.

[53] S. Harnad, The symbol grounding problem, Physica D 42 (1990) 335–346.

[54] M. Krotzsch, F. Simancik, I. Horrocks, A description logic primer, arXiv:1201.4089v3.

[55] D. McDermott, PDDL – the Planning Domain Definition Language, Tech. rep., Yale (1998).

[56] `https://github.com/EMAROLab/OWL-ROSPlan/tree/master/rosplan_knowledge_base/`, [Online; accessed May 18 2018] (2017).

[57] M. Fox, D. Long, Modelling mixed discrete-continuous domains for planning, Journal of Artificial Intelligence Research 27 (2006) 235–297.

[58] `https://github.com/EMAROLab/paco_actions`, [Online; accessed May 18 2018] (2017).

[59] `http://wiki.ros.org/ar_track_alvar`, [Online; accessed May 18 2018] (2016).

[60] `https://github.com/EMAROLab/ARMOR`, [Online; accessed May 18 2018] (2017).

[61] A. Capitanelli, M. Maratea, F. Mastrogiovanni, M. Vallati, Automated planning techniques for robot manipulation tasks involving articulated objects, in: Proceedings of the 2017 Conference of the Italian Association for Artificial Intelligence (AIxIA 2017), Bari, Italy, 2017.

[62] F. Mastrogiovanni, A. Paikan, A. Sgorbissa, Semantic-aware real-time scheduling in robotics, IEEE Transactions on Robotics 29 (1) (2013) 118–135.

[63] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, The Description Logic handbook: theory, implementation, and applications, Cambridge University Press, New York, NY, USA, 2003.

[64] M. Smith, C. Welty, D. McGuinness, OWL web ontology language guide, `https://www.w3.org/TR/owl-guide/`.

[65] I. Horrocks, P. Patel-Schneider, S. Bechhofer, D. Tsarkov, Owl rules: a proposal and prototype implementation, Journal of Web Semantics 3 (1) (2005) 23–40.

Appendix 1: Representation Models in the Ontology

An ontology $\Sigma = (TBox, ABox)$ is a 2-ple where the $TBox$ is a terminological taxonomy of axioms storing definitions of classes and relationships within a domain, and the $ABox$ is an assertional taxonomy representing the related factual knowledge. In planHRC, both $TBox$ and $ABox$ are described using the Description Logic formalism [63] through its computational variant Web Ontology Language (OWL), and in particular OWL-DL [64], plus SWRL rules for deductive reasoning [65].

The taxonomy in the $TBox$ models *types* used by the *Planner* module to process PDDL descriptions as primitive classes derived from $\mathsf{Type} \sqsubseteq \top$, e.g.,

$\mathsf{Link} \sqsubseteq \mathsf{Type}$, $\mathsf{Joint} \sqsubseteq \mathsf{Type}$, $\mathsf{Orientation} \sqsubseteq \mathsf{Integer}$. Relevant *predicates* are modelled as classes derived from $\mathsf{Predicate} \sqsubseteq \top$. For instance, $\mathsf{Connected}$ is used to relate a Joint to a Link, as:

$$\begin{aligned} \mathsf{Connected} \sqsubseteq \mathsf{Predicate} \sqcap \\ &\exists \mathsf{arg1.Joint} \sqcap =_1 \mathsf{arg1} \sqcap \\ &\exists \mathsf{arg2.Link} \sqcap =_1 \mathsf{arg2}. \end{aligned} \tag{8}$$

Arguments $\mathsf{arg1}$ and $\mathsf{arg2}$ relate *one* Joint with *one* Link. Intermediate links are modelled using two $\mathsf{Connected}$ predicates, for the downstream and upstream links, respectively. It is necessary to specify the orientation associated with a Link with respect to a Joint:

$$\begin{aligned} \mathsf{HasOrientation} \sqsubseteq \mathsf{Predicate} \sqcap \\ &\exists \mathsf{arg1.Joint} \sqcap =_1 \mathsf{arg1} \sqcap \\ &\exists \mathsf{arg2.Orientation} \sqcap =_1 \mathsf{arg2}, \end{aligned} \tag{9}$$

where the semantics of $\mathsf{arg2}$ depends on whether we adopt absolute or relative angles. In the planning process, an orientation can take values in the set $\mathcal{O}$, with the aim of reducing the state space involved in the planning process. The set $\mathcal{O}$ is represented as a collection of predicates relating pairwise values:

$$\begin{aligned} \mathsf{OrientationOrd} \sqsubseteq \mathsf{Predicate} \sqcap \\ &\exists \mathsf{arg1.Orientation} \sqcap =_1 \mathsf{arg1} \sqcap \\ &\exists \mathsf{arg2.Orientation} \sqcap =_1 \mathsf{arg2}. \end{aligned} \tag{10}$$

For instance, if only two possible orientations are allowed, namely 30 *deg* and 45 *deg*, $\mathcal{O}$ can be modelled using only *one* predicate $\mathsf{OrientationOrd(ord\_30\_45)}$ such that:

$$\begin{aligned} &\mathsf{arg1(ord\_30\_45, 30)}, \\ &\mathsf{arg2(ord\_30\_45, 45)}, \end{aligned} \tag{11}$$

where it is intended that orientations 30 *deg* and 45 *deg* are associated with $\mathsf{arg1}$ and $\mathsf{arg2}$, respectively. Other predicates are described in a similar way.

Conditional operators in PDDL are modelled in the $TBox$ using *conditional predicates* to be mapped to PDDL operators:

$$\begin{aligned}\mathsf{CondPredicate} \sqsubseteq \mathsf{Predicate} \sqcap \\ &\exists \mathsf{forall}.\mathsf{Type} \sqcap \geq_1 \mathsf{forall} \sqcap \\ &\exists \mathsf{when}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{when} \sqcap \\ &\exists \mathsf{eff-\_a}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{eff-\_a}, \sqcap \\ &\exists \mathsf{eff+\_a}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{eff+\_a},\end{aligned} \tag{12}$$

where the intuitive meaning is that for all Type individuals specified in the relationship, when specific Predicate individuals hold, the additional effects must be considered.

We define $\mathsf{Action} \sqsubseteq \top$ as:

$$\begin{aligned}\mathsf{Action} \sqsubseteq \top \sqcap \\ &\exists \mathsf{params}.\mathsf{Type} \sqcap \geq_1 \mathsf{params} \sqcap \\ &\exists \mathsf{pre}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{pre} \sqcap \\ &\exists \mathsf{eff-}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{eff-} \sqcap \\ &\exists \mathsf{eff+}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{eff+} \sqcap \\ &\exists \mathsf{condEff}.\mathsf{CondPredicate}.\end{aligned} \tag{13}$$

In (13), we do not assume the presence of a relationship condEff to the aim of modelling both actions and conditional actions using the same definition. In our $TBox$, two actions are defined, namely $\mathsf{RotateClockwise} \sqsubseteq \mathsf{Action}$, and $\mathsf{RotateAntiClockwise} \sqsubseteq \mathsf{Action}$.

One predicate used as part of conditional effects is Affected, which models how changing a link orientation propagates via connected upstream or downstream joints:

$$\begin{aligned}\mathsf{Affected} \sqsubseteq \mathsf{Predicate} \sqcap \\ &\exists \mathsf{arg1}.\mathsf{Joint} \sqcap =_1 \mathsf{arg1} \sqcap \\ &\exists \mathsf{arg2}.\mathsf{Link} \sqcap =_1 \mathsf{arg2} \sqcap \\ &\exists \mathsf{arg3}.\mathsf{Joint} \sqcap =_1 \mathsf{arg3},\end{aligned} \tag{14}$$

which states that a change in orientation related to the joint in arg1 is affected by rotations of joints specified in arg3, as obtained when operating on the link in arg2.

A *state* s (perceived, current, predicted or expected) is represented as a set of predicates:

$$\begin{aligned}\mathsf{State} \sqsubseteq\ & \top \sqcap \\ & \exists \mathsf{madeof}.\mathsf{Predicate} \sqcap \geq_1 \mathsf{madeof},\end{aligned} \tag{15}$$

through the relationship madeof, which must include at least one Predicate for the state to be formally expressed. A planning *problem* is modelled as having an initial and a goal State:

$$\begin{aligned}\mathsf{Problem} \sqsubseteq\ & \top \sqcap \\ & \exists \mathsf{init}.\mathsf{State} \sqcap =_1 \mathsf{init} \sqcap \\ & \exists \mathsf{goal}.\mathsf{State} \sqcap =_1 \mathsf{goal}.\end{aligned} \tag{16}$$

Finally, a Plan $\sqsubseteq \top$ is made up of actions:

$$\begin{aligned}\mathsf{Plan} \sqsubseteq\ & \top \sqcap \\ & \exists \mathsf{madeof}.\mathsf{Action} \sqcap \geq_1 \mathsf{madeof}.\end{aligned} \tag{17}$$

On-line, the *ABox* is updated each time a new image is acquired by the *Perception* module, and maintains descriptions in the form of assertions. Let us describe what happens at each iteration with an example. If the robot perceived an object configuration like the one in Figure 2 on the top, four Link instances:

$$\mathsf{Link(l1)} \quad \mathsf{Link(l2)} \quad \mathsf{Link(l3)} \quad \mathsf{Link(l4)} \tag{18}$$

and four Joint instances:

$$\mathsf{Joint(j1)} \quad \mathsf{Joint(j2)} \quad \mathsf{Joint(j3)} \quad \mathsf{Joint(j4)} \tag{19}$$

are used to represent it. The object's structure is modelled as a description including the set of predicate instances:

$$\begin{aligned}&\mathsf{Connected}(\mathsf{connected\_j1\_l1})\\&\mathsf{Connected}(\mathsf{connected\_j2\_l1})\\&\mathsf{Connected}(\mathsf{connected\_j2\_l2})\\&\dots\\&\mathsf{Connected}(\mathsf{connected\_j4\_l4}).\end{aligned} \tag{20}$$

where $\mathsf{connected\_j1\_l1}$ is such that:

$$\begin{aligned}&\mathsf{arg1}(\mathsf{connected\_j1\_l1}, \mathsf{j1})\\&\mathsf{arg2}(\mathsf{connected\_j1\_l1}, \mathsf{l1}),\end{aligned} \tag{21}$$

as specified in (8). Other *Predicate* instances can be generated in a similar way. Assuming that $\theta_1^a = 45\ deg$, $\theta_2^a = 330\ deg$, $\theta_3^a = 30\ deg$ and $\theta_4^a = 315\ deg$, orientations are represented as:

$$\begin{aligned}&\mathsf{HasOrientation}(\mathsf{has\_orientation\_j1\_45})\\&\mathsf{HasOrientation}(\mathsf{has\_orientation\_j2\_330})\\&\mathsf{HasOrientation}(\mathsf{has\_orientation\_j3\_30})\\&\mathsf{HasOrientation}(\mathsf{has\_orientation\_j4\_315}),\end{aligned} \tag{22}$$

where, focusing on $\mathsf{arg2}$ only:

$$\begin{aligned}&\mathsf{arg2}(\mathsf{has\_orientation\_j1\_45}, 45)\\&\mathsf{arg2}(\mathsf{has\_orientation\_j2\_330}, 330)\\&\mathsf{arg2}(\mathsf{has\_orientation\_j3\_30}, 30)\\&\mathsf{arg2}(\mathsf{has\_orientation\_j4\_315}, 315),\end{aligned} \tag{23}$$

All such $\mathsf{Connected}$ and $\mathsf{HasOrientation}$ instances contribute to the definition of the current state $\mathsf{State(state\_c)}$ by means of a set of assertions like:

$$\begin{aligned}&\mathsf{madeof}(\mathsf{state\_c}, \mathsf{connected\_j1\_l1})\\&\mathsf{madeof}(\mathsf{state\_c}, \mathsf{connected\_j2\_l1})\\&\dots\\&\mathsf{madeof}(\mathsf{state\_c}, \mathsf{has\_orientation\_j4\_315}, 315),\end{aligned} \tag{24}$$

as foreseen by (15). Similar descriptions for problems and, after the planning process occurs, plans, can be introduced as well.

When a new goal state $\mathsf{State(state\_g)}$ is encoded in the ontology, a new $\mathsf{Problem(problem\_c)}$ is created, such that, according to (16):

$$\begin{aligned}&\mathsf{init(problem\_c, state\_c)}\\&\mathsf{goal(problem\_c, state\_g)},\end{aligned} \tag{25}$$

and the *Planner* module is activated. A translation process generates the proper PDDL formulation by querying the $TBox$ (to generate the PDDL *domain*) and the $ABox$ (to generate the PDDL *problem*). Each class in the $TBox$ roughly corresponds to a section of the domain, whereas $\mathsf{state\_c}$ and $\mathsf{state\_g}$ in the $ABox$ define the initialisation and goal sections of a problem.

After a plan has been found and validated (see Section 4.3), each action is encoded back in the ontology as an instance of Action, and therefore all relationships param, pre, $\mathsf{eff}-$ and $\mathsf{eff}+$ are specified in terms of Type and $\mathsf{Predicate}$ instances. If an action has conditional effects, also $\mathsf{condEff}$ is determined. As a consequence, a set of intermediate expected states is create as:

$$\begin{aligned}&\mathsf{State(state\_e\_1)}\\&\mathsf{State(state\_e\_2)}\\&\dots\\&\mathsf{State(state\_e\_n + 1)}\end{aligned} \tag{26}$$

as described in Section 4.1. In particular, $\mathsf{state\_e\_1} \equiv \mathsf{state\_c}$, $\mathsf{state\_e\_n} + 1 \equiv \mathsf{state\_g}$, and the intermediate expected states are generated using (6) and (7). When State individuals are generated, the *Execution* module is activated and the human-robot cooperation process can start.