Epistemic Multiagent Reasoning with Collaborative Robots

Davide Soldà[1,2,*], Francesco Fabiano[1,3,*] and Agostino Dovier[1,4]

[1]*Constraint and Logic Programming Lab, University of Udine, 33100 Udine, Italy.* http://clp.dimi.uniud.it/
[2]*Embedding System unit of Fondazione Bruno Kessler, 38123 Trento, Italy*
[3]*Department of Mathematical, Physical and Computer Sciences, University of Parma, 43124 Parma, Italy*
[4]*Department of Mathematics, Computer Science and Physics, University of Udine, 33100 Udine, Italy*

Abstract

Over the last few years, the fields of Artificial Intelligence, Robotics and IoT have gained a lot of attention. This increasing interest has brought, among other things, to the development of autonomous multi-agent systems where robotic entities may interact with each other. As for all the other autonomous settings, also these systems require arbitration. Our work tries to address this problem by presenting a framework that embeds both a classical and a multi-agent epistemic (epistemic, for brevity) planner in a robotic control architecture. The idea is to combine the *(i)* classical and the *(ii)* epistemic solvers to model efficiently the interaction with: the *(i)* physical world and the *(ii)* information flows, respectively. In particular, the presented architecture starts by planning on the "epistemic level" refining then single-agent world-altering actions thanks to the classical planner. To further optimize the solving process, we also introduce the concept of macros in epistemic planning. Macros, in fact, have been successfully employed in classical planning as they allow for opportune aggregations of actions that may lead to a reduction of plans' length. Finally, the overall framework is exemplified and validated with two Franka Emika manipulators. This allowed us to empirically justify how the combination of the two planning approaches (classical and epistemic), and the introduction of macros, reduce the computational time required by the orchestrating phase.

Keywords

Multi-Agent Planning, Epistemic Reasoning, Answer Set Programming, Robot Operating System

1. Introduction

In the recent years, the field of cognitive robotics experienced significant progress, both from the academic (see [1] for a detailed survey) and the industrial, *e.g.* [2], [3], [4], point of view. Even if most of current autonomous systems frameworks are designed for scenarios where only a single entity interacts with the environment, also multi-agent settings are well studied [5], [6]. However, there is not focus on modeling robots beliefs in goals or action pre/post conditions.

In particular, we envisioned and developed e-PICO (**EPI**stemic Reasoner **C**ollaborative R**O**bots): a planning framework that coordinates a set of autonomous agents. This setup is able

CILC 2022: 37th Italian Conference on Computational Logic, June 29 – July 1, 2022, Bologna, Italy
*Corresponding author.
solda.davide@spes.uniud.it (D. Soldà); francesco.fabiano@unipr.it (F. Fabiano); agostino.dovier@uniud.it (A. Dovier)
0000-0002-1161-0336 (F. Fabiano); 0000-0003-2052-8593 (A. Dovier)

to reason on complex multi-agent epistemic concepts while dealing with their computational issues, that arise from the inherent complexity of epistemic reasoning. This is achieved by combining techniques from the Multi-agent Epistemic Planning (MEP) field and the more efficient classical planning approaches. Thanks to this combination our architecture can benefit from the generality derived by MEP solvers and the efficiency of the classical ones. That is, e-PICO consider both of an epistemic and a classical description of the planning problem and combines the two solving techniques in a hierarchical way. As we will see later (Section 4) in much greater detail, the architecture firstly solves the problem with a greater level of abstraction using the epistemic solver. Then, the classical resolution process is used to refine the world-altering actions suggested by the epistemic counterpart.

Finally, inspired by [7], we also propose the concept of *macros* in the MEP environment. The use of macros helps in reducing the burden of the epistemic planning process, allowing e-PICO to address multi-agent robotics planning scenarios. In fact, such domains present several complications, *e.g.*, the number of possible actions, that if not addressed correctly could render the solving process unfeasible.

To the best of our knowledge, both *i)* the combination of the epistemic and a classical solving processes to model a robotic environment with epistemic goals; and *ii)* the use of macros in the multi-agent epistemic planning context are original contributions of this work.

2. Action Languages and Planning

The area of *automated planning* is one of the most prominent in Artificial Intelligence. This branch studies how to devise tools that help us in deciding the best course of actions to reach a given goal, an activity that is done continuously in our life. Automated planning, therefore, represents one of the most interesting aspect of AI and, consequently, has been vastly studied [8, 9, 10]. Even if we often need to make decisions based on our beliefs, about the environments and about others' beliefs, automated planners usually do not consider such intricacy. In fact, most of the efforts in the planning community address domains where the concept of beliefs and/or knowledge is not taken into account. Nonetheless, the growing interest in AI is pushing researchers to steadily improve and to model more realistic scenarios. This momentum brought, among other things, to the formalization of a far more compelling (w.r.t. more classical approaches) form of planning, that is the so-called *Epistemic Multi-Agent Planning* (MEP). This area of planning reasons within environments where the streams of "knowledge" or "beliefs" need to be considered (see [11, 12] for a more detailed introduction to the topic).

2.1. Multi-Agent Epistemic Planning

Formalizing and reasoning on the idea of knowledge and beliefs has always been of great interest among various research fields (*e.g.*, philosophy, logics, and computer science). In particular, in 1962, Hintikka proposed the first complete axiomatization of these concepts [13]. From this initial effort stemmed the field of *epistemic/doxastic logic* which aims to formalize and reason on information. While this area of research presents various challenges, it focused only on capturing the knowledge relations in static domains. To represent even more interesting and realistic scenarios *Dynamic Epistemic Logic* (DEL) was introduced; that is the logic of

reasoning on information flows in dynamic domains where agents can act and alter these relations themselves.

DEL represents the foundation of Multi-Agent Epistemic Planning, the setting concerned with finding the best series of action, that modifies the information flows, to reach goals that (might) refer to agents' knowledge/beliefs. In what follows, for brevity, we will use the term "knowledge" to encapsulate *both* the notions of an agent's knowledge and beliefs. In fact, these concepts are captured by the same modal operator in DEL and their difference resides in structural properties that the epistemic states respect (see [11] for more details). As it is not the objective of this paper to completely present MEP, in what follows we will provide only some fundamental concepts that are necessary to explain the contribution of this work. Far more complete introductions to this topic may be found in [11, 14, 15].

Let us start by presenting the language of well-formed DEL formulae used to express agents' knowledge. This is expressed as follows:

$$\varphi ::= \mathsf{f} \mid \neg\varphi \mid \varphi \wedge \psi \mid \mathbf{B}_{\mathsf{i}}\varphi \mid \mathbf{C}_{\alpha}\varphi,$$

where f is a propositional atom called a *fluent*, i is an agent that belongs to the set of agents $\mathcal{AG}$ s.t. $|\mathcal{AG}| \geq 1$, φ and ψ are belief formulae and $\emptyset \neq \alpha \subseteq \mathcal{AG}$. A *fluent formula* is a DEL formula with no occurrences of modal operators. A *belief formula* is recursively defined as follows:

- A fluent formula is a belief formula;
- If φ is a belief formula and $\mathsf{i} \in \mathcal{AG}$, then $\mathbf{B}_{\mathsf{i}}\varphi$ ("i knows/believes that φ") is a belief formula where the modal operator $\mathbf{B}$ captures the concept of *knowledge*;
- If φ_1, φ_2 and φ_3 are belief formulae, then $\neg\varphi_3$ and $\varphi_1 \ \mathsf{op} \ \varphi_2$ are belief formulae, where $\mathsf{op} \in \{\wedge, \vee, \Rightarrow\}$;
- If φ is a belief formula and $\emptyset \neq \alpha \subseteq \mathcal{AG}$ then $\mathbf{C}_{\alpha}\varphi$ is a belief formula where $\mathbf{C}_{\alpha}$ captures the *Common knowledge* of the set of agents α.

The formula $\mathbf{C}_{\alpha}\varphi$ translates intuitively into the conjunction of the following belief formulae:

- every agent in α knows φ;
- every agent in α knows that every agent in α knows φ;
- and so on *ad infinitum*.

The semantics of DEL formulae is traditionally expressed using *pointed Kripke structures* [16], but also other representations are possible [17, 18]. We refer the interested reader to [11, 12, 17] for a comprehensive introduction to how Kripke structures, or similar formalisms, are used to capture the idea of an epistemic state and how the concept of entailment is defined.

Let us note that the epistemic action language that we will consider in our work implements three *types of action* and three *observability relations*. These are standard concepts in the epistemic planning community and, therefore, we will provide an intuitive description of those addressing the interested reader to [14] for a complete description of this topic. In particular, we assume that each agent can execute one of the following types of action:

- *World-altering* action (also called *ontic*): used to modify certain properties (*i.e.*, fluents) of the world.
- *Sensing* action: used by an agent to refine her beliefs about the world.
- *Announcement* action: used by an agent to affect the beliefs of other agents.

Moreover, each agent is associated to one of the following observability relations during an action execution:

- *Fully-observant*: the agent is aware of the action execution and also knows the effect of the action.
- *Partially-observant*: the agent is aware of the action execution without knowing the effects of the action. Let us note that no agent can be partially observant of an ontic action as it is impossible to decouple the witnessing of a world-altering action and the witnessing of its effects.
- *Oblivious*: the agent is not even aware of the action execution.

Each type of action defines a transition function and alter an epistemic state in different ways. Given the complexity of the topic and the space limitations we address the reader to [14, 18, 19] for a formal definition of these, and others, update functions on diverse epistemic state representations.

Finally, let us introduce the concept of *MEP domain* that, intuitively, contains the information needed to describe a planning problem in a multi-agent epistemic setting.

Definition 2.1 (MEP Domain)**.** *A* multi-agent epistemic planning domain *is a tuple* $\mathcal{D} = \langle \mathcal{F}, \mathcal{AG}, \mathcal{A}, \varphi_{ini}, \varphi_{goal} \rangle$*, where* $\mathcal{F}$*,* $\mathcal{AG}$*,* $\mathcal{A}$ *are the sets of* fluents*,* agents*,* actions *of* $\mathcal{D}$*, respectively;* φ_{ini} *and* φ_{goal} *are DEL formulae that must be entailed by the* initial *and* goal e-state*, respectively. The former e-state describes the domain's initial configuration while the latter encodes the desired one.*

We refer to the elements of a domain $\mathcal{D}$ with the parenthesis operator; *e.g.*, the fluent set of $\mathcal{D}$ is denoted by $\mathcal{D}(\mathcal{F})$. An *action instance* $\mathsf{a}\langle \mathsf{i} \rangle \in \mathcal{D}(\mathcal{AI}) = \mathcal{D}(\mathcal{A}) \times \mathcal{D}(\mathcal{AG})$ identifies the execution of action a by an agent i. Let $\mathcal{D}(\mathcal{S})$ be the set of all possible e-states of the domain. The *transition function* $\Phi : \mathcal{D}(\mathcal{AI}) \times \mathcal{D}(\mathcal{S}) \to \mathcal{D}(\mathcal{S}) \cup \{\emptyset\}$ formalizes the semantics of action instances (the result is the empty set if the action instance is not executable).

2.2. The $\mathcal{A}^V$ language

In this section we briefly introduce the $\mathcal{A}^V$ action description language, which has been designed to be integrated into e-PICO. It essentially corresponds to language $\mathcal{A}$, but some typed variables have been introduced (see [20] as a reference for the nomenclature of planning languages). Typed variables are merely used to write schemes describing finite sets of causal laws that are formed according to the same pattern. Schemes are particularly useful in some application

domains, such as when an action that models an agent's movement can be parameterized by a variable that can be instantiated with a series of different positions.

In e-PICO the $\mathcal{A}^V$ planning instances are translated into an Answer Set Programming ASP instances. ASP is one the most prominent logic programming paradigms and it is particularly useful in knowledge-intensive applications, see Gelfond and Lifschitz [21] for an introduction to its semantic. Its use as a planning framework, has been deeply explained in [22]. For space reasons we omit the correctness proof of the compiler from $\mathcal{A}^V$ to ASP.

The choice of encoding the planning problems in ASP, already presented in [23], is justified by the fact that ASP as a planning engine outperforms standard task planner for PDDL, where *i)* domains are rich in fluents and *ii)* plans are usually short [24]. Such characteristics precisely characterize how we want to use the classical planner in the robotics domain use case.

3. Collaborative Robots

3.1. Robot Operating System

Robot Operating System (ROS) is the standard *de facto* framework for developing robotic applications. It consists of a communication framework for sending messages between processes. The ROS integrated executable programs are instantiated as special processes, known as nodes, and are organized in packages. A ROS package might contain ROS nodes, a ROS-independent library, a dataset, configuration files, a third-party piece of software, or anything else that logically constitutes a useful module. There are several packages which are directly provided by ROS and which can be used by user-defined programs. One package that is particularly used in this work is `MoveIt!`, which is widely used for motion planning and execution in the robotics community.

3.2. The Robots Franka Emika

The framework here presented has been validated in a simple multi-agent scenario in which two Franka Emika robots have been involved. Franka Emika is a robotic arm with 7 Degrees Of Freedom (DOF) with torque sensors at each joint. It is also equipped with a gripper that allows the automated "arm" to handle objects.

The client side of the Franka Control Interface is called `libfranka`. At a higher level we find `franka_ros`, which is a ROS package that contains the description of the robot and the end-effector in terms of kinematics, joint limits, visual surfaces and collision space, an hardware abstraction of the robot for the ROS control framework based on the `libfranka` API, and a set of services to expose the full `libfranka` API in the ROS ecosystem. Let us note that, although the target robots are two Franka Emika, the architecture can be adapted to another collection of manipulators with only minor changes.

4. The e-PICO System

4.1. Macros in Multi-Agent Epistemic Planning

As already mentioned, a significant contribution of this work is the formalization, and consequent employment, of *macros* in the MEP setting. A *macro*, that can be informally described as "an encapsulated sequences of elementary planning operators", is formally defined as follows:

Definition 4.1 (Macro). *Let* $\mathcal{D}(\mathsf{i}), \mathcal{D}(\mathsf{j}) \in \mathcal{D}(\mathcal{AI})$ *be two action instances, and* $\mathsf{s} \in \mathcal{D}(\mathcal{AG})$ *be an e-state of a given a domain* $\mathcal{D}$*. A macro* $\mathsf{m}_{\mathsf{i},\mathsf{j}} \in \mathcal{D}$ *representing the subsequent execution of* j *after* i *can be defined as* $\Phi(\mathsf{j}, \Phi(\mathsf{i}, \mathsf{s}))$*. Let us note that* i) *we assume that if action* a *is not executable in a state* s*, then the result of the update* $\Phi(\mathsf{a}, \mathsf{s})$ *is* $\bot$*; and* ii) *the execution of any action* a *over* $\bot$ *results in* $\bot$ *as well.*

The introduction of macros is justified by the fact that, often, patterns of actions performed in sequence are repeated in the same domain. For example, it often happens that an ontic action is followed by an announcement action, because an agent may want to communicate the results of the former to another (oblivious) agent. For now, as pointed out in the Dagstuhl seminar [25], there have been many proposals in the classification of epistemic actions. As pointed out above, the community mostly agrees that the basic classification considers *ontic*, *announcement* and *sensing* as possible categories. Among these three types of actions, only the ontic one has some effect on the physical world. That is why we consider the renaming two, *i.e.*, sensing and announcement, as *purely epistemic* actions. Consequently, we say that a task is purely epistemic if it involves only announcement or sensing actions, or a macro aggregating these two type. In Listing 1 the function call `is_pure_epistemic_task(task)` checks exactly this property.

4.2. The Architecture

In this section, we describe the general architecture and functionality of the e-PICO system, that is the main contribution of this work. e-PICO provides high-level knowledge representation and planning capabilities to ROS-based autonomous robots via the action description languages $\mathcal{A}^V$ (Section 2) and E-PDDL [26]. The main object of our framework is to provide a tool that supports multi-agent epistemic planning in the robotics setting. This is accomplished through a combination of MEP and classical planning solving techniques. In particular, e-PICO combines these two strategies in a hierarchical way. To be more precise, our architecture firstly employs multi-agent epistemic planning for generating tasks, then, at lower level, it exploits a classical planner to break down tasks into simpler actions. Finally, e-PICO directly converts the results of the planning processes into `MoveIt!` commands that can be then executed by the robots. Let us note that we employ an hierarchical combination of the two solving techniques (*i.e.*, classical and epistemic) to avoid unfeasibility in the planning process for domains rich in fluents and actions. The key idea is, in fact, to abstract most of the domain intricacy from the epistemic level and handle it at the classical level, when it is possible, given its vastly superior performances.

Let us now explain in greater detail how the e-PICO system processes, solves and executes a planning problem in the robotic environment. To better visualize the overall framework, we present a graphical representation of it in Figure 1. First of all the *master process* reads the domain descriptions, that contain the initial state description for both the MEP and classical

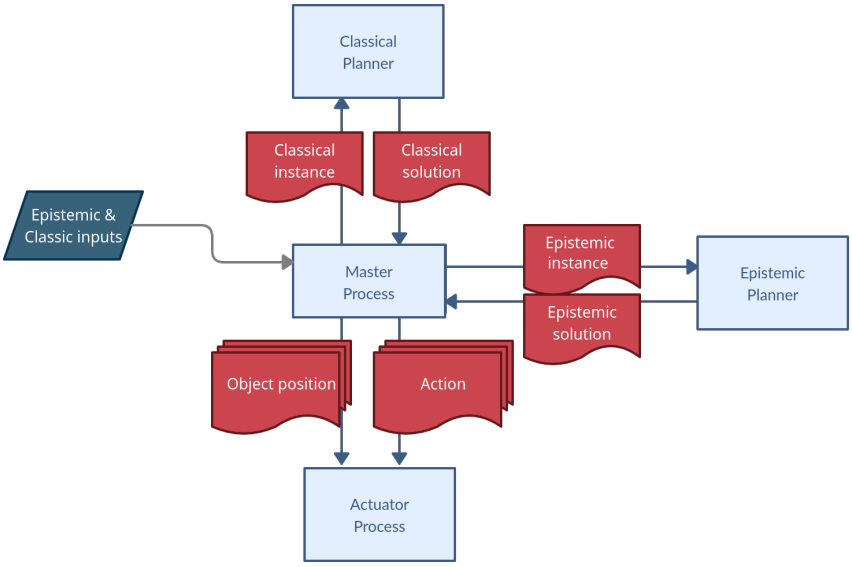


Figure 1: A class-based representation of the e-PICO system.

setting. At this point it is able to send messages to the *actuator process* regarding the objects that need to be considered by the motion planner in order to avoid collisions. Once these messages arrive to the *actuator process*, it adds them to the MoveIt! planning scene.

Then, it sends the epistemic planning instance, specified in E-PDDL [26], to an epistemic planner, *i.e.*, EFP [18], which is employed as a black box and returns the sequence of epistemic tasks to be executed. After this first resolution, each task is properly processed following Listing 1. The first step is to break down (*to flat*) macros, then we can reason on a sequence of no macro tasks, hereinafter called *simple tasks*. Intuitively, a simple pure epistemic task can be directly executed. Simple ontic tasks that alter the physical world will undergo further processing. To break down ontic simple tasks into a sequence of classical actions, the *master process* defines an instance of classical planning where the initial and the goal states are defined to match the ontic simple task's conditions and effects. Then, iteratively, the *master process* sends the classical action tasks to the *actuator process*, which translate them into MoveIt! commands, and makes finally the robots execute the movement.

Listing 1: pseudo-code for the main steps of e-PICO.

```
def process_e_plan(list<task> task_plan, c_init_state c_init):
  for task in task_plan:
    if is_macro_task(task):
      simple_task_plan = break_down_macro(macro=task)
      for simple_task in simple_task_plan:
        c_init = process_simple_task(t, c_init)
    else:
      c_init = process_simple_task(task)
```

```
def process_simple_task(task t, c_init_state c_init):
  # invariant: not is_macro_task(t)
  if is_epistemic_task(task=t): # sensing or announcement
    execute(pure_epistemic_task=t)
  else:
    c_plan, c_init = send_to_c_planner(ontic_task=t, initial_state=
        c_init)
    for c_action in c_plan:
      send_to_actuator(c_action)
  return c_init
```

4.3. Modeling the Problem Instance

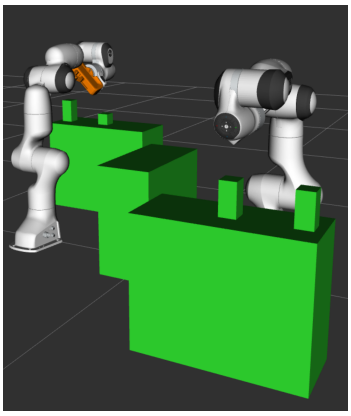
When we model a scenario as a planning problem in order to eventually effectively execute it, we need to address the "anchoring" problem. I.e., we need to connect the model description to the physical objects in the real world. Furthermore, in the e-PICO approach, the user also needs to choose what should be modelled at the epistemic level and what at the classical one, the latter in fact should provide low-level actions that are easy to interface with the MoveIt! API. Let us take the process of grasping, executed by an automated arm, as an example. It is composed of different phases: i) *pre-grasping*, in which the *end-effector* approaches the object to be grasped with a given direction and orientation. ii) actual *grasping*, in which the final joints of the gripper close. Finally, iii) *post-grasping*, in which the end-effector moves away from the position in which it grasped the object in a given direction and orientation. During each of these phases process we must make sure that the robotic arm does not collide with the objects in its workspace. Therefore, MoveIt! is provided with information about location of fixed objects to calculate a trajectory without collisions with fast algorithms, such as *Rapidly-exploring Random Tree*. Furthermore, the aspects of pre-grasping and post-grasping are hidden in the planning model, that sees the grasping action as atomic, the control of the three-phases of grasping has been hard-coded in the *actuator process*.

4.4. Case study

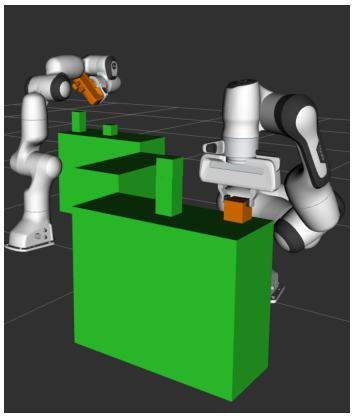
We can now present an application of e-PICO in a two agents scenario. The diverse problem instances contain several fluents and actions, as usual in the robotics domain.

The two agents involved are two robotic arms, called robot1 and robot2. In front of each of them, three stacks of colored blocks lie on a table. Initially, the they do not know what blocks they are able to manipulate (Figure 2a presents an example of initial state). To collect such information, they can perform sensing actions to see which blocks they initially have in front of them. Robots can also take a block (as depicted in Figure 2b) to then place it on the *shared table*, as in Figure 2c. We assume that only once a robot has placed the block on the shared table, it is able to communicate the color of such a block to the other arm. Examples in Figure 2 are obtained using the RViz simulator [27] integrated in MoveIt!.

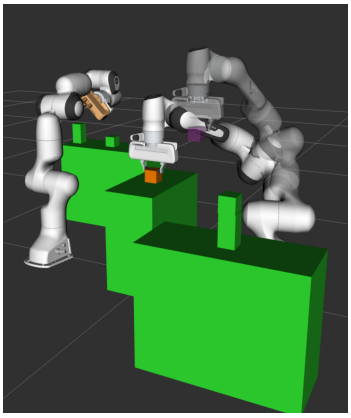
For the sake of readability, we will not report all the actions' descriptions, rather we will show only some meaningful examples. Initial state and the fluents encoding are not presented, because they just follow the standard PDDL notation [28, 29]. Let us start, by showing an ontic



(a) Initial state.



(b) A robot picking up a block from its table.



(c) A robot placing a block on the shared table.

Figure 2: Examples of the robotic test environment states. A video of the "real" execution with the Panda Robots is available: http://clp.dimi.uniud.it/sw/

action in Listing 2. This action encodes the idea that if an agent beliefs that it is holding a block of a certain color and the shared table is free, then, after the execution, the agent will occupy the shared table with the aforementioned block (and will know about this). For a detailed explanantion on the syntax of E-PDDL we address the reader to [26]

Listing 2: Example of ontic action.

```
(
 :action          pick_and_place_on_shared
 :act_type        ontic
 :parameters      (?ag ?ag2 - agent ?c ?c2 - color)
 :precondition    (and    ([?ag](hold ?ag ?c)) (not (occup_by ?ag2 ?c2)))
 :effect          (and (occup_by ?ag ?c) (hold ?ag ?c))
 :observers       (?ag)
)
```

Similarly, for instance, an annoucement action can be executed by the agent that has just moved the block, announcing the newly verified property to the other arm.

Listing 3: Example of annoucement action.

```
(
 :action          announce_block_on_barrier
 :act_type        announcement
 :parameters      (?ag - agent)
 :precondition    (and (occup_by ?ag ?c) (hold ?ag ?c) ([?ag](occup_by ?
     ag ?c)) ([?ag](hold ?ag ?c)))
 :effect          (and (occup_by ?ag ?c) (hold ?ag ?c))
 :observers       (forall (?ag2 - agent) (?ag2))
)
```

We can now apply the 4.1 definition to actions Listings 2 and 3 with the following macro as result, see Listing 4.

Listing 4: Example of a *macro* aggregating an ontic and an announcing action.

```
(
 :action         announce_pick_and_place_on_shared
 :act_type       ontic
 :parameters     (?ag ?ag2 - agent ?c ?c2 - color)
 :precondition   (and    ([?ag](hold ?ag ?c)) (not (occup_by ?ag2 ?c2)))
 :effect         (and (occup_by ?ag ?c) (hold ?ag ?c))
 :observers      (forall (?ag2 - agent) (?ag2))
)
```

Let us now consider what happens to an epistemic ontic action that is refined by the classical planner. Consider a configuration in which the a red block is stacked under a black block and the epistemic ontic action suggests to move the red block on the shared table. The robot cannot take directly the red block, but it should firstly move the black one at the top of another stack and then take the red block to put it on the shared table. Therefore such an epistemic ontic task may be translated into the following sequence of actions:

Listing 5: List of single-arm actions.

```
% pick the black block over the red one from stack 2, pos 1
pick(black,red,1,2)

% place the black block over the green one in stack 1, pos 2
place(black,green,2,1)

% pick the red block from stack 2,pos 0
pick(red,table,0,2)

% place the red block on the shared table
put_on_barrier(red)
```

5. Experimental results

To further asses and demonstrate the capabilities of the developed framework, we tested our architecture on a wide spectrum of scenarios that stem from the configuration described in Section 4.4. Experiments showed the positive impact of using macros in the MEP setting. Furthermore, we were also able to prove the feasibility of the solving procedure in real-world situations.

All the experiments have been conducted on Intel i7-8565U CPU at 1.80GHz and Ubuntu 18.04 OS. For each problem instance, a time limit of 110 seconds was applied. Clingo [30] was used to solve the ASP encoding of $\mathcal{A}^V$, and EFP [18] to tackle the resolution of MEP problems.

First let us highlight how the use of macros can help in reducing the solving time, in Table 1. Test-cases are obtained with a cross product between goal with an increasing difficulty and

with an increasing number of blocks inserted in the domain (rows and columns of Table 1 respectively). In particular, we have that:

α: has the objective to let one agent know the color of at least one block initially located in the table in front of the other robot;

β: has same goal as of α while also requesting that the shared table must be free at the end of the plan;

γ: encodes the scenario where the goal is for a robot to know two different colors of blocks initially placed in front of the other arm;

δ: shares the goal of γ with an "extra" condition imposing that the shared table must be free when the planning process is concluded.

For the sake of readability we will make use of the following notations in Table 1:

- ND, that stands for Not Defined. This is used, for example, to indicate that instance γ cannot be performed with just one color as it required that a robot learns two different colors while only one is available.
- TO stands for Time Out. As said before, after 110 seconds the planning procedure is forcefully stopped if it could not find a solution.

From the domain it is evident that whenever an agent takes a block from the shared table, to place it on top of one of its stacks, announcing that the table has been freed right after is very convenient. This small sequence of actions constitutes the macro `announce_pick_and_place_on_shared`. Finally, in Table 1, we compare the solving process when this macro is not activated ("Macro no") and when it is ("Macro yes"). In summary, it is evident that macros improve considerably performance. Their use makes it possible to have better scalability both in terms of number of fluents and of plan lengths.

	1 Color		**2** Colors		**3** Colors		**4** Colors	
Macro	no	yes	no	yes	no	yes	no	yes
α	0.019	**0.004**	1.200	**0.080**	5.192	**1.382**	104.000	**25.889**
β	0.083	**0.013**	4.946	**0.242**	21.780	**4.235**	TO	**85.144**
γ	ND	ND	18.663	**1.427**	TO	**24.805**	TO	TO
δ	ND	ND	TO	**4.827**	TO	**92.998**	TO	TO

Table 1
Time, in seconds, to find a goal, given an initial state. Each instance varies the number of available colors. In **boldface** is highlighted the fastest solving process w.r.t. the activation of the *pspb_and_announce* macro. Let us note that all the values were obtained by averaging the times of 5 iterations on the same instance.

For the sake of completeness, let us now address the initial state generation. It is the first step performed during the calculation of the espitemic plan. This process is empirically almost independent of the use of macros, but it is affected by the number of fluents and actions used in

the planning model. To better exemplify this, let us report the average times needed to compute the initial states, increasing the number of fluents, in our case the number of colored blocks: **1)** for one color 0.003 s; **2)** for two colors 0.031 s; **3)** for three colors 0.126 s; and **4)** for four colors 2.117 s. As a consequence of this observation, it is a good rule of thumb, to limit the number of fluents considered to the strict necessary required for the epistemic planning.

Finally, as a design choice we decided to run the classical planner at run-time, each time its use is required to break down an ontic task in a sub-plan of classical actions. Justification for this choice is provided by Table 2, in which, for each instance, the cumulative time required by the A^V solver and the times it was called are reported. Let us note that we did not report the times required by the epistemic planner without the support of the classical one, *i.e.*, the times required by the solver if the domain was entirely described at the epistemic level. The reason is that, without the assistance of the "breakdown" procedure, provided by the classical solver, the solving time always reached the timeout.

	1 Color		**2** Colors		**3** Colors		**4** Colors	
	time	calls	time	calls	time	calls	time	calls
α	0.074	1	0.083	1	0.092	1	0.112	1
β	0.157	2	0.172	2	0.177	2	0.202	2
γ	ND	ND	0.250	3	0.267	3	TO	TO
δ	ND	ND	0.332	4	0.351	4	TO	TO

Table 2
Cumulative time, in seconds, to break down ontic tasks into classical actions. The number of calls to the classical planner is reported in the column "calls".

6. Related Work and Conclusions

While the field of cognitive robotics has made significant progress over the last few years, there are still some opening questions regarding how to integrated new AI components. Moreover, multi-agent scenarios where the acting entities can perform low-level sensing and control tasks are becoming more and more available both in the industrial and academic environments. And, at the same time, epistemic planning is receiving a lot of interest.

In [31] and in [32] Capitanelli *et al.* propose a set of PDDL+ formulations that allows to model the problem of manipulating articulated objects in a three-dimensional workspace with a dual-arm robot. Instead, [7] address the same domains while encoding the planning problem directly in ASP making a strong use of macros. Another similar tool that inspired our research is the ROSoClingo [33] ROS package. This framework integrates Clingo [30] into the ROS service and `actionlib` architecture, providing an high-level ASP-based interface to control the behavior of a robot. While these works provided the foundation for our research and are far more complete tools, they do not consider neither the information flows between agents, nor their knowledge. This aspect is key in every multi-agent scenario, where reasoning on the perspective of others should be taken into consideration.

That is why we proposed `e-PICO`, a framework that offers the possibility of modeling

multi-agent epistemic planning problems in robotics environments. This tool stems from the aforementioned approaches and integrates the latest MEP techniques to tackle the planning problems considering also the information flows. While integrating the epistemic aspect of multi-agent domain is, in our opinion, of the utmost importance, it requires high computational resources. That is why, e-PICO also introduces two methods to improve the planning times and to have feasible solving processes: **i)** a hierarchical usage of epistemic and classical planners to abstract and simplify the problems when considered by epistemic solvers; and **ii)** the employment of macros in epistemic planning.

Acknowledgments

The research presented in this paper is partially supported by Fondazione Friuli/Università di Udine project on *Artificial Intelligence for Human Robot Collaboration*, by INdAM-GNCS Projects NoRMA and CUP_E55F22000270001.

References

[1] M. Bhatt, E. Erdem, F. Heintz, M. Spranger, Cognitive robotics, 2016.
[2] S. Tonetta, ROBDT robotic digital twin, https://es.fbk.eu/index.php/projects/robdt/, 2021. Accessed: 2022-05-10.
[3] M. Bozzano, R. Bussola, M. Cristoforetti, M. Jonas, K. Kapellos, A. Micheli, D. Soldà, S. Tonetta, C. Tranoris, A. Valentini, Robdt: Ai-enhanced digital twins for space exploration robotic assets, in: The 2022 International Conference on Applied Intelligence and Informatics (AII2022), 2022.
[4] A. K. Jónsson, N. Muscettola, P. H. Morris, K. Rajan, Next generation remote agent planner, 1999.
[5] T. Niemueller, T. Hofmann, G. Lakemeyer, Goal reasoning in the clips executive for integrated planning and execution, 2019.
[6] T. Hofmann, T. Viehmann, M. Gomaa, D. Habering, T. Niemueller, G. Lakemeyer, C. Team, Multi-agent goal reasoning with the clips executive in the robocup logistics league., in: ICAART (1), 2021, pp. 80–91.
[7] R. Bertolucci, A. Capitanelli, C. Dodaro, N. Leone, M. Maratea, F. Mastrogiovanni, M. Vallati, An asp-based framework for the manipulation of articulated objects using dual-arm robots, in: International Conference on Logic Programming and Nonmonotonic Reasoning, Springer, 2019, pp. 32–44.
[8] M. Helmert, The fast downward planning system, Journal of Artificial Intelligence Research 26 (2006) 191–246.
[9] N. Lipovetzky, H. Geffner, Best-first width search: Exploration and exploitation in classical planning, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA, 2017, pp. 3590–3596. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14862.
[10] S. Richter, M. Westphal, The lama planner: Guiding cost-based anytime planning with

landmarks, Journal of Artificial Intelligence Research 39 (2010) 127–177. doi:10.1613/jair.2972.
[11] R. Fagin, J. Y. Halpern, Reasoning about knowledge and probability, Journal of the ACM (JACM) 41 (1994) 340–367. doi:10.1145/174652.174658.
[12] T. Bolander, A gentle introduction to epistemic planning: The del approach, arXiv preprint arXiv:1703.02192 (2017).
[13] J. Hintikka, Knowledge and Belief: An Introduction to the Logic of the Two Notions, Ithaca: Cornell University Press, 1962.
[14] C. Baral, G. Gelfond, E. Pontelli, T. C. Son, An action language for multi-agent domains: Foundations, arXiv preprint arXiv:1511.01960 (2015).
[15] H. Van Ditmarsch, W. van Der Hoek, B. Kooi, Dynamic epistemic logic, volume 337, Springer Science & Business Media, 2007.
[16] S. A. Kripke, Semantical considerations on modal logic, Acta Philosophica Fennica 16 (1963) 83–94.
[17] J. Gerbrandy, W. Groeneveld, Reasoning about information change, Journal of Logic, Language and Information 6 (1997) 147–169. doi:10.1023/A:1008222603071.
[18] F. Fabiano, A. Burigana, A. Dovier, E. Pontelli, EFP 2.0: A multi-agent epistemic solver with multiple e-state representations, in: Proceedings of the 30th International Conference on Automated Planning and Scheduling, 2020, pp. 101–109.
[19] F. Fabiano, A. Burigana, A. Dovier, E. Pontelli, T. C. Son, Multi-agent epistemic planning with inconsistent beliefs, trust and lies, in: D. N. Pham, T. Theeramunkong, G. Governatori, F. Liu (Eds.), PRICAI 2021: Trends in Artificial Intelligence - 18th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2021, Hanoi, Vietnam, November 8-12, 2021, Proceedings, Part I, volume 13031 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 586–597. URL: https://doi.org/10.1007/978-3-030-89188-6_44. doi:10.1007/978-3-030-89188-6_44.
[20] M. Gelfond, V. Lifschitz, Action languages, Electron. Trans. Artif. Intell. 2 (1998) 193–210. URL: http://www.ep.liu.se/ej/etai/1998/007/.
[21] V. Lifschitz, Answer set programming, Springer Berlin, 2019.
[22] A. Dovier, A. Formisano, E. Pontelli, Perspectives on logic-based approaches for reasoning about actions and change, in: M. Balduccini, T. C. Son (Eds.), Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning - Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday, volume 6565 of *Lecture Notes in Computer Science*, Springer, 2011, pp. 259–279. URL: https://doi.org/10.1007/978-3-642-20832-4_17. doi:10.1007/978-3-642-20832-4_17.
[23] A. Burigana, F. Fabiano, A. Dovier, E. Pontelli, Modelling multi-agent epistemic planning in ASP, Theory Pract. Log. Program. 20 (2020) 593–608. URL: https://doi.org/10.1017/S1471068420000289. doi:10.1017/S1471068420000289.
[24] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, P. Stone, Task planning in robotics: an empirical comparison of pddl-and asp-based systems, Frontiers of Information Technology & Electronic Engineering 20 (2019) 363–373.
[25] C. Baral, T. Bolander, H. van Ditmarsch, S. McIlrath, Epistemic Planning (Dagstuhl Seminar 17231), Dagstuhl Reports 7 (2017) 1–47. URL: http://drops.dagstuhl.de/opus/volltexte/2017/8285. doi:10.4230/DagRep.7.6.1.

[26] F. Fabiano, B. Srivastava, J. Lenchner, L. Horesh, F. Rossi, M. B. Ganapini, E-PDDL: A standardized way of defining epistemic planning problems, CoRR abs/2107.08739 (2021). URL: https://arxiv.org/abs/2107.08739. arXiv:2107.08739.

[27] A. H. C. Robert Haschke, Chris Lalancette, Rviz documentation, http://wiki.ros.org/rviz/UserGuide, ???? Accessed: 2022-02-24.

[28] D. McDermott, M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, D. Weld, PDDL - The Planning Domain Definition Language, Technical Report, Technical Report, 1998.

[29] M. Fox, D. Long, Pddl2.1: An extension to pddl for expressing temporal planning domains, Journal of Artificial Intelligence Research 20 (2003) 61–124. URL: http://dx.doi.org/10.1613/jair.1129. doi:10.1613/jair.1129.

[30] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot ASP solving with clingo, CoRR abs/1705.09811 (2017).

[31] A. Capitanelli, M. Maratea, F. Mastrogiovanni, M. Vallati, Automated planning techniques for robot manipulation tasks involving articulated objects, in: Conference of the Italian Association for Artificial Intelligence, Springer, 2017, pp. 483–497.

[32] A. Capitanelli, M. Maratea, F. Mastrogiovanni, M. Vallati, On the manipulation of articulated objects in human–robot cooperation scenarios, Robotics and Autonomous Systems 109 (2018) 139–155.

[33] B. Andres, P. Obermeier, O. Sabuncu, T. Schaub, D. Rajaratnam, Rosoclingo: A ros package for asp-based robot control, arXiv preprint arXiv:1307.7398 (2013).