Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning
Main Track

DEL-based Epistemic Planning for Human-Robot Collaboration: Theory and Implementation

Thomas Bolander , Lasse Dissing , Nicolai Herrmann
Technical University of Denmark
{tobo, ldiha}@dtu.dk, nicolai@herrmann.dk

Abstract

Epistemic planning based on Dynamic Epistemic Logic (DEL) allows agents to reason and plan from the perspective of other agents. The framework of DEL-based epistemic planning thereby has the potential to represent significant aspects of Theory of Mind in autonomous robots, and to provide a foundation for human-robot collaboration in which coordination is achieved implicitly through perspective shifts. In this paper, we build on previous work in epistemic planning with implicit coordination. We introduce a new notion of indistinguishability between epistemic states based on bisimulation, and provide a novel partition refinement algorithm for computing unique representatives of sets of indistinguishable states. We provide an algorithm for computing implicitly coordinated plans using these new constructs, embed it in a perceive-plan-act agent loop, and implement it on a robot. The planning algorithm is benchmarked against an existing epistemic planning algorithm, and the robotic implementation is demonstrated on human-robot collaboration scenarios requiring implicit coordination.

1 Introduction

One important application of epistemic planning is Human-Robot Collaboration, that is, multiple robots and humans collaboratively working towards a joint goal. The inclusion of human collaborators greatly increases the complexity, due to the difficulties of human-robot communication (Thomaz, Hoffman, and Cakmak 2016). It is therefore desirable to build systems capable of *implicit coordination*, i.e., coordination based upon shared information about the world and the ability to take the perspective of the others.

Consider a scenario with a human and a service robot, where the human says: "I am looking for my glasses. Can you help me find them?". Assuming the robot adopts the goal to help, it becomes a *joint goal* between the human and the robot to locate the glasses. It is an *epistemic* goal: the goal is for the human to *know* where the glasses are. The robot might come up with the following sequential plan: *findLocationOfGlasses*, *announceLocationOfGlasses*. While this plan works correctly in the nominal case, it could lead to socially awkward behaviour if the human finds the glasses herself first. The robot, detecting the glasses in her hands, might then announce: "I found your glasses! They are in your hands!". This could happen if the robot is not able to take her perspective and understand that if she picks up the glasses herself, it must imply that she already knows where they are (and hence that the goal has been reached).

In this paper, we develop a planning algorithm capable of taking the perspective of humans and reason about their knowledge and uncertainty, hence avoiding this kind of social awkwardness. The typical approach used by prior human-robot collaboration planners is to maintain a distinct knowledge base for each agent in the scenario (Talamadupula et al. 2014; Buisan and Alami 2021). However, such first-order representations are not expressive enough to handle more complex scenarios involving nested perspective-taking (Lemaignan and Dillenbourg 2015).

For this reason, we instead approach this problem through the theoretical framework of *epistemic planning with implicit coordination* proposed by Engesser et al. (2017) and further detailed by Bolander et al. (2018). Using Dynamic Epistemic Logic (DEL), they represent communication and other epistemic actions in the same action representation language as regular ontic actions. Furthermore, they suggest finding *policies* (mapping states to actions) instead of sequential plans in order to handle non-deterministic execution and uncertainty about the plans chosen by other agents. Planners built upon this framework have been described in several prior works. The original paper by Engesser et al. (2017) briefly describes an implementation using a breadth-first search over an AND-OR graph. Engesser and Miller (2020) describe a planner which compiles a decidable fragment of epistemic planning into *fully observable nondeterministic planning*. Reifsteck et al. (2019) describe an algorithm and implementation capable of finding epistemic planning policies using a Monte-Carlo Tree Search over epistemic states.

We continue this line of work by making theoretical improvements to the framework (Section 3–4): an improved notion of indistinguishability based on bisimulation and a novel partition refinement algorithm for computing unique representatives of indistinguishable states. Furthermore, we describe in detail an AND-OR graph search algorithm for finding implicitly coordinated policies (Section 6). Additionally, we implement a planner based upon the algorithm, and benchmark it against prior work (Section 7). We also integrate the planner with a robotics system, and test it on human-robot collaboration scenarios (Section 7). The Supplementary Material referred to in the paper can be found at

https://github.com/Zeltex/Implicit_Coordination.

2 Dynamic Epistemic Logic

The version of DEL used in this paper is the version used by Bolander et al. (2018), which is essentially classic DEL (Baltag, Moss, and Solecki 1998) extended to allow *postconditions* and *multi-pointed models*. Hence all definitions in this section can also be found in Bolander et al. (2018), except with minor differences in the notational and syntactic conventions. Additionally, for technical simplicity, in epistemic states and actions, we require all worlds and events to be reachable from the designated worlds and events. We take our propositional atoms to be a set $P(c_1, \ldots, c_n)$ of ground atoms of first-order predicate logic (Bolander 2017; Dissing and Bolander 2020). Let Ψ be a set of predicates of first-order logic, $\mathcal{O}$ a finite set of *objects*, and $\mathcal{A}$ a finite set of *agents*. Our epistemic language $\mathcal{L}(\Psi, \mathcal{O}, \mathcal{A})$, often abbreviated $\mathcal{L}$, is then given by $\phi ::= \top \mid \bot \mid P(c_1, \ldots, c_n) \mid \neg\phi \mid \phi \wedge \phi \mid K_i\phi \mid C\phi$, where $i \in \mathcal{A}$, $P \in \Psi$ is a predicate of arity $n \in \mathbb{N}$ and $c_i \in \mathcal{O} \cup \mathcal{A}$ for all i. The set of atoms of the form $P(c_1, \ldots, c_n)$ is denoted $Atm(\Psi, \mathcal{O}, \mathcal{A})$, often abbreviated Atm. $K_i\phi$ is read as "agent i knows ϕ" and $C\phi$ as "it is common knowledge that ϕ". Formulas of the form $C\phi$ are called *common knowledge formulas*.

Definition 1. An *(epistemic) state* is $s = (W, R, L, W_d)$, where W is a non-empty finite set of *worlds*, $R\colon \mathcal{A} \to \mathscr{P}(W \times W)$ assigns to each agent $i \in \mathcal{A}$ an equivalence relation R_i called an *accessibility relation*; $L\colon W \to \mathscr{P}(Atm)$ assigns to each world w a *labelling* (the set of atoms true in w); and $W_d \subseteq W$ is the set of *designated worlds*. We will require that all worlds are reachable from the set of designated worlds, i.e., for each $w \in W$ there is a $w_d \in W_d$ such that $(w_d, w) \in (\cup_{i \in \mathcal{A}} R_i)^*$. When $W_d = \{w_d\}$ for some $w_d \in W$, the state is called *global* and we then often write (W, R, L, w_d) for $(W, R, L, \{w_d\})$. In this case, w_d is called the *actual world*. For any state $s = (W, R, L, W_d)$, let $Globals(s) = \{(W, R, L, w) \mid w \in W_d\}$. A state s that is not global can be considered as a compact representation of $Globals(s)$, the set of global states it contains.

The truth of $\mathcal{L}$-formulas in epistemic states is defined as follows, with standard semantics for the propositional cases:

$$
\begin{array}{lll}
(W,R,L,W_d) \models \phi & \text{iff} & \text{for all } w \in W_d, (W,R,L,w) \models \phi \\
(W,R,L,w) \models p & \text{iff} & p \in L(w) \quad \text{where } p \in Atm \\
(W,R,L,w) \models K_i\phi & \text{iff} & \text{for all } v \in W, \text{if } (w,v) \in R_i \\
 & & \text{then } (W,R,L,v) \models \phi \\
(W,R,L,w) \models C\phi & \text{iff} & \text{for all } v \in W, \\
 & & \text{if } (w,v) \in (\bigcup_{i \in \mathcal{A}} R_i)^* \text{ then } (W,R,L,v) \models \phi
\end{array}
$$

Given a global state $s = (W, R, L, w_d)$ and an agent i, the accessibility relation R_i models the *run-time uncertainty* of agent i. In s, agent i cannot distinguish any of the worlds v with $(w_d, v) \in R_i$ from the actual world w_d, i.e., the knowledge of i is given by what is true in all of those worlds v. Higher-order knowledge is represented via sequences of relations, e.g. "agent i knows that agent j knows ϕ" is true in s iff $s \models K_iK_j\phi$, which holds when ϕ is true in all worlds accessible from w_d by the composite relation $R_i; R_j$. We use

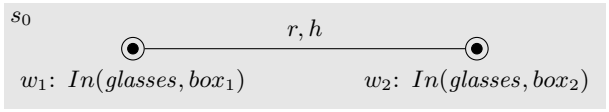

Figure 1: The initial state $s_0 = (\mathcal{M}, \{w_1, w_2\})$ of Example 1. The worlds are represented as nodes, the accessibility relations as labelled edges (reflexive loops omitted) and an interior dot ⦿ marks designated worlds.

the set W_d of worlds to model *plan-time uncertainty* (Bolander and Andersen 2011), that is, uncertainty caused by nondeterministic actions, such as a coin toss, where the exact outcome is not revealed until the action is performed at runtime. A state (W, R, L, W_d) is a *local state* for agent i if W_d is closed under R_i. The *perspective shift* of a state $s = (W, R, L, W_d)$ with respect to an agent i is defined as $s^i = (W, R, L, \{v \mid (w, v) \in R_i \text{ and } w \in W_d\})$. The perspective-shifted state s^i represents the plan-time uncertainty of agent i in state s. If s is a global state, s^i is the view (or perspective) of i on state s. The state $(s^i)^j$, for some agent j, is then j's perspective on i's perspective on the state s, and it encodes what j knows about what i knows about s (Engesser et al. 2017).

Example 1. Consider a version of the "missing glasses" scenario from the introduction with $\mathcal{A} = \{r, h\}$, where r is the robot and h the human. We assume that initially both of them know that the glasses are in one of two boxes, box_1 or box_2, but they do not know which, modelled by s_0 of Figure 1 which is local to both agents. World w_j, $j = 1, 2$, represents the possibility that the glasses are in box_j. The relations encode that both agents consider both worlds possible: each $i \in \mathcal{A}$ knows the glasses are in one of the boxes, $s_0 \models K_i(In(glasses, box_1) \vee In(glasses, box_2))$, but does not know which, $s_0 \models \neg K_i In(glasses, box_1) \wedge \neg K_i In(glasses, box_2)$.

Definition 2. An *(epistemic) action* of agent i is an expression of the form $i{:}a$ with $a = (E, Q, pre, post, E_d)$ where E is a non-empty finite set of *events*; $Q\colon \mathcal{A} \to \mathscr{P}(E \times E)$ assigns to each agent $j \in \mathcal{A}$ an equivalence relation Q_j called an *accessibility relation*; $pre\colon E \to \mathcal{L}$ assigns a *precondition* to each event; $post\colon E \to \mathcal{L}$ assigns a *postcondition* to each event with the restriction that for all $e \in E$, $post(e)$ is a conjunction of literals (atoms and their negations) with each atom occurring at most once; and $E_d \subseteq E$ is a subset of *designated events*. We will require that E_d is closed under Q_i.

An epistemic action $i{:}a$ is an action executed by agent i (we can read $i{:}a$ as "i does a"). When $a = (E, Q, pre, post, E_d)$, E_d represents the plan-time uncertainty of agent i concerning which event will occur. Since we require E_d to be closed under Q_i, it means an action $i{:}a$ is not only executed by i, but also seen from the perspective of i. An action with $|E_d| > 1$ is thus seen as non-deterministic from the perspective of agent i. Applying an action to a state is achieved through the *product update* defined next.

Definition 3. Let $s = (W, R, L, W_d)$ be a local epistemic state for agent i and $i{:}a = i{:}(E, Q, pre, post, E_d)$ an epis-

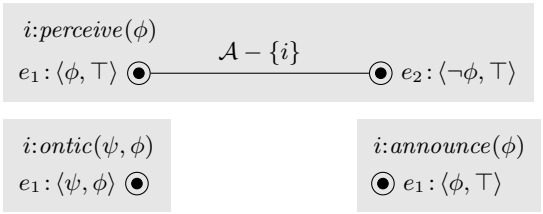

Figure 2: The generic actions used in the "missing glasses" scenario. Each event $e \in E$ is labelled by $\langle pre(e), post(e) \rangle$. i:$perceive(\phi)$ models that agent i semi-privately observes whether ϕ holds. i:$announce(\phi)$ models a public announcement of ϕ and i:$ontic(\psi, \phi)$ models a public ontic action with ψ as precondition and ϕ as effect, e.g. h:$ontic(In(cube, box_1), \neg In(cube, box_1) \wedge In(cube, box_2))$ represents agent h moving the cube from box 1 to box 2.

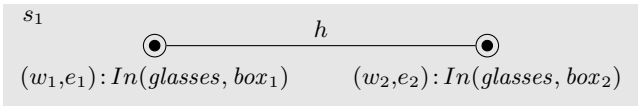

Figure 3: The state s_1 resulting from the product update of s_0 from Figure 1 with r:$perceive(In(glasses, box1))$.

temic action of i. The *product update* of s with i:a is defined as the epistemic state $s \otimes i{:}a = (W', R', L', W'_d)$ where

- $W'_d = \{(w, e) \in W_d \times E_d \mid (W, R, L, w) \vDash pre(e)\}$
- $R'_j = \{((w, e), (v, f)) \mid (w, v) \in R_j \text{ and } (e, f) \in Q_j\}$
- $W' = \{(v, f) \in W \times E \mid (W, R, L, v) \vDash pre(f) \text{ and } ((w, e), (v, f)) \in (\cup_{i \in \mathcal{A}} R'_i)^* \text{ for some } (w, e) \in W'_d\}$
- $L'((w, e)) = \{p \in Atm \mid \text{either } post(e) \vDash p \text{ or both } (W, R, L, w) \vDash p \text{ and } post(e) \nvDash \neg p\}$

Furthermore, i:a is said to be *applicable* in s if for all $w \in W_d$ there exists an event $e \in E_d$ s.t. $(W, R, L, w) \models pre(e)$.

Note that as a consequence of our conventions, the product update $s \otimes i{:}a$ will always be a local state for agent i.

Example 2. Extending Example 1, for each agent i and formula ϕ we can define an action i:$perceive(\phi)$, provided in Figure 2, modelling that agent i semi-privately senses whether ϕ holds, e.g. if $\phi = In(glasses, box_1)$ and $i = r$, it models that the robot r checks whether the glasses are in box_1. Note that all agents except i will not get to know whether ϕ was true or not, so in the example with $\phi = In(glasses, box_1)$, it corresponds to agent i peeking into box_1 in the presence of the other agents but without the other agents seeing what i sees. Letting s_0 be the state from Figure 1, we can compute the result $s_1 = s_0 \otimes r$:$perceive(In(glasses, box_1))$ as seen in Figure 3. r no longer has any (run-time) uncertainty regarding the location of the glasses, i.e. $s_1 \vDash K_r In(glasses, box_1) \vee K_r In(glasses, box_2)$. h still doesn't know their location, but h now knows that r knows their location, $s_1 \vDash K_h(K_r In(glasses, box_1) \vee K_r In(glasses, box_2))$.

3 Uniformity and Bisimulations

In general in automated planning and game theory, a *policy* (or *strategy*) is a mapping from states to actions. In our case, each agent i has its own policy, which is then a mapping from states into actions of agent i. In planning under partial observability and in games with imperfect information, agent i might not be able to distinguish certain states. We thus assume we are given an *indistinguishability relation* $\sim_i$ for agent i on the set of states. Given $\sim_i$ for agent i, a policy π_i for agent i is called *uniform* if whenever $s \sim_i s'$ we have $\pi_i(s) = \pi_i(s')$. Thus an agent must make the same choice in states it can't distinguish between. This is essential for a policy to make sense from the subjective perspective of the planning agent (Jamroga and Aagotnes 2007). What is the relevant notion of indistinguishability on epistemic states? Well, if two local states of agent i are modally equivalent (satisfy the same formulas), then necessarily they must be indistinguishable to that agent (the agent has exactly the same knowledge in the two states). So in our setting, the uniformity condition reduces to this: if s and s' are modally equivalent local states for agent i then $\pi_i(s) = \pi_i(s')$. In order to ensure uniformity of our policies, we hence need a method to check pairs of states for modal equivalence. For standard single-pointed epistemic states (global states), it is well-known that modal equivalence coincides with bisimilarity (Blackburn and van Benthem 2007). For multi-pointed epistemic states as we consider here, a notion of bisimulation was defined by Bolander and Andersen (2011), however that definition does not ensure that modal equivalence coincides with bisimilarity (the condition on how the designated worlds are related by the bisimulation relation is too weak). We here fix the definition and prove that it gives the expected correspondence. Nothing in this section is going to rely on the accessibility relations R_i being equivalence relations, so the proofs will also go through for multi-pointed **K** models (epistemic logic with arbitrary accessibility relations).

Definition 4. A *bisimulation* between epistemic states $s = (W, R, L, W_d)$ and $t = (W', R', L', W'_d)$ is a binary relation $Z \subseteq W \times W'$ satisfying:

[atom] If $(w, w') \in Z$, then $L(w) = L'(w')$.

[forth] If $(w, w') \in Z$ and $(w, v) \in R_i$, then there exists v' such that $(v, v') \in Z$ and $(w', v') \in R'_i$.

[back] If $(w, w') \in Z$ and $(w', v') \in R'_i$, then there exists v such that $(v, v') \in Z$ and $(w, v) \in R_i$.

[designated] If $w \in W_d$, then there exists a $w' \in W'_d$ with $(w, w') \in Z$, and vice versa.

Two epistemic states s and t are called *bisimilar* iff there exists a bisimulation between them, and we then write $s \leftrightarrows t$.

Proposition 1. Two epistemic states are bisimilar iff they are modally equivalent (satisfy the same set of formulas).

Proof sketch. The trick is to replace multi-pointed models by single-pointed models as follows: The set of designated worlds of the original model is replaced by a single fresh actual world, and all the old designated worlds are made accessible from this new actual world by the accessilibity relation of a fresh agent. By thus translating multi-pointed models into single-pointed models, we can rely on the standard proof of the correspondence between bisimilarity and modal equivalence (Blackburn, de Rijke, and Venema 2001). See Supplementary Material for the full proof. □

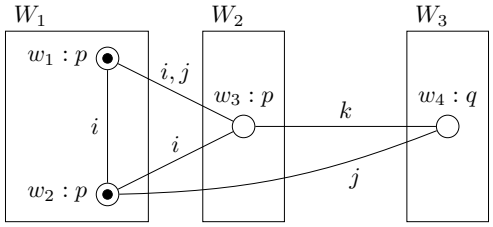

Figure 4: An epistemic state and a partition (W_1, W_2, W_3) of its worlds.

Given the result above, we can guarantee uniformity of a policy by checking that it returns the same set of actions on any pair of bisimilar states. We can accomplish this, and at the same time keep the policies minimal, by defining unique representatives of each class of indistinguishable states and then only define the policy on these representatives. There already exists a notion of *bisimulation contraction* of an epistemic state (Blackburn and van Benthem 2007), which can be computed using a partition refinement algorithm (Aceto, Ingólfsdóttir, and Srba 2012). However, using partition refinement to contract a set of bisimilar states does not guarantee a *unique* representative, but may produce any state from a set of isomorphic states, i.e., a set of states which are identical except for the world names. We will define our own partition refinement algorithm called *OrderedPartitionRefinement* which indeed guarantees finding a unique representative of each class of modally equivalent/bisimilar states. It does so by simply assuming a total order on all the relevant syntactical constructs, and using it to determine the order of the individual steps of the algorithm as well as the order of the blocks within a partition. This ensures that for bisimilar states, the same steps are taken in the same order, resulting in the same worlds in the same order. To our knowledge, this is the first such algorithm guaranteeing unique representatives, and may hence be of independent interest within epistemic logic and labelled transition systems.

4 Ordered Partition Refinement

An *ordered partition* of a set W is a tuple $(W_1, \ldots, W_n)$ with $W_1 \cup \cdots \cup W_n = W$ and all W_i being non-empty and pairwise disjoint. The elements W_i of the partition are called *blocks* and n is called the *length* of the partition. The *index* of an element $w \in W$ in a partition $(W_1, \ldots, W_n)$ is the unique i for which $w \in W_i$. From now on we will assume that W is the set of worlds of some epistemic state. Figure 4 shows an epistemic state together with an ordered partition $P = (W_1, W_2, W_3)$ of length 3. The *signature* of a world w wrt. a partition $(W_1, \ldots, W_n)$ is the set $\sigma_{(W_1,\ldots,W_n)}(w) = L(w) \cup \{(i,k) \in \mathcal{A} \times \mathbb{N} \mid \text{for some } v,\ (w,v) \in R_i \text{ and } v \in W_k\}$, e.g. the signature of w_2 in Figure 4 wrt. the partition (W_1, W_2, W_3) is $\{p, (i,1), (i,2), (j,3)\}$. Given a partition P, we say that a signature σ *occurs in* or *is contained in* a set of worlds W' if there exists $w \in W'$ with $\sigma_P(w) = \sigma$. From now on, we will assume a fixed total ordering on signatures (e.g. induced by the alphabet used for naming predicates, objects and agents). Our ordered partition refinement algorithm is presented as Algorithm 1.

Algorithm 1: *OrderedPartitionRefinement*

Input: An epistemic state $s = (W, R, L, W_d)$
Output: A contracted state $s' = (W', R', L', W'_d)$

1 $(\sigma_1, \ldots, \sigma_k) \leftarrow$ the signatures of the worlds in W wrt. the trivial partition (W)—listed in increasing order
2 **for** $i \leftarrow 1$ **to** k **do** `/* initial partition */`
3 | $W_i \leftarrow \{w \in W \mid \sigma(w) = \sigma_i\}$
4 $newlength \leftarrow k$
5 **repeat** `/* begin refinement step */`
6 | $oldlength \leftarrow newlength$
7 | **for** $i \leftarrow 1$ **to** $oldlength$ **do**
8 | | $(\sigma_1, \ldots, \sigma_l) \leftarrow$ the unique signatures in W_i wrt. the partition $(W_1, \ldots, W_{newlength})$—listed in increasing order
9 | | **for** $j \leftarrow 2$ **to** l **do**
| | | `/* we split block` W_i `by moving its` σ_j `worlds to a new block */`
10 | | | $W_{newlength+1} \leftarrow \{w \in W_i \mid \sigma(w) = \sigma_j\}$
11 | | | $W_i \leftarrow W_i - \{w \mid \sigma(w) = \sigma_j\}$
12 | | | $newlength \leftarrow newlength + 1$
13 **until** $newlength = oldlength$
14 $W' \leftarrow \{1, \ldots, newlength\}$
15 **for each** $i \in \mathcal{A}$ **do**
16 | $R'_i \leftarrow \{(m,n) \mid (w,v) \in R_i, w \in W_m, v \in W_n\}$
17 **for** $j \leftarrow 1$ **to** $newlength$ **do**
18 | $L'(j) \leftarrow L(w)$ for any $w \in W_j$
19 $W'_d \leftarrow \{m \mid W_m \cap W_d \neq \emptyset\}$
20 **return** (W', R', L', W'_d)

Let's consider how the algorithm would work on the epistemic state of Figure 4. The signatures of the trivial partition $P = (W)$ are $\sigma_P(w_1) = \sigma_P(w_2) = \{p, (i,1), (j,1)\}$, $\sigma_P(w_3) = \{p, (i,1), (j,1), (k,1)\}$ and $\sigma_P(w_4) = \{q, (j,1), (k,1)\}$. So the initial partition constructed in lines 1–3 will be $P' = (\{w_1, w_2\}, \{w_3\}, \{w_4\}) = (W_1, W_2, W_3)$ (or one of its permutations, depending on the total order on the signatures). In the first iteration ($i = 1$) of the for-loop in lines 7–12, we will get $(\sigma_1, \sigma_2) = (\sigma_{P'}(w_1), \sigma_{P'}(w_2)) = (\{p, (i,1), (i,2), (j,2)\}, \{p, (i,1), (i,2), (j,3)\})$ (or in opposite order). This means that in lines 10–12 we split the block W_1 into two blocks: A reduced W_1 block containing only the worlds with signature σ_1 and a new block W_4 containing the worlds with signature σ_2. I.e. we get $W_1 = \{w_1\}$ and $W_4 = \{w_2\}$. The next two iterations of the for-loop will not change anything, as W_2 and W_3 are already singletons. Since this refinement step increased the length of the partition, the repeat-until loop will go through another iteration. In this iteration nothing gets split, and the algorithm will finally return the contracted model with worlds $W' = \{1, \ldots, 4\}$ based on the final partition. In the contracted model, the world named 2 is constructed from the second block $W_2 = \{w_3\}$ of the final partition. Hence world 2 represents w_3 of the original model and inherits the properties of w_3 (its label and outgoing edges). Note that in this particular example, no worlds become identified (put into

the same block in the final partition). The algorithm will simply end up reshuffling the order of the worlds. This happens since none of the four worlds are bisimilar.

Lemma 1. Let Z be a bisimulation between epistemic states $s = (W, R, L, W_d)$ and $t = (W', R', L', W'_d)$. Suppose that we concurrently execute $OrderedPartitionRefinement$ on both s and t. The following invariant holds at the beginning of each iteration of the repeat-until loop: If $(w, w') \in Z$ then w and w' have the same index in their respective partitions.

Proof. The proof is by induction on the number of iterations of the repeat-until loop. We will use $P = (W_1, \ldots, W_n)$ to denote the current partition of s at the beginning of the relevant iteration of the loop, and $P' = (W'_1, \ldots, W'_{n'})$ to denote the corresponding partition of t. The base case is at the beginning of the first iteration, i.e., after having constructed the initial partition. Suppose $(w, w') \in Z$. It follows from the definition of bisimulations that w and w' have the same labels (by [atom]) and the same agent names on their outgoing edges (by [back] and [forth]). Hence they must have the same signature wrt. the trivial partitions ((W) and (W'), respectively). By the construction in lines 1–3, it follows that w and w' have the same index in the initial partitions.

For the induction step, suppose the invariant holds at the beginning of the Nth iteration of the repeat-until loop. We then show it still holds at the end of this iteration.

Claim 1: For any world $w \in W$ there is a world $w' \in W'$ with $(w, w') \in Z$ and vice versa. Proof: Suppose $w \in W$. Since s and t are bisimilar and have all worlds reachable from a designated world (Definition 1), it follows from [designated], [forth] and [back] that there must exist $w' \in W'$ such that $(w, w') \in Z$. The other direction is symmetric.

Claim 2: At the beginning of the Nth iteration, if $(w, w') \in Z$ then w and w' have the same signatures wrt. the current partitions. We want to show $\sigma_P(w) = \sigma_{P'}(w')$. We do this by showing that each element of $\sigma_P(w)$ is also an element of $\sigma_{P'}(w')$ (the other direction being symmetric). For the elements of $L(w)$ this holds by [atom], since $(w, w') \in Z$. Consider then an element $(i, k) \in \sigma_P(w)$. This means there exists a world $v \in W_k$ such that $(w, v) \in R_i$. By [forth], there must exist a world v' such that $(v, v') \in Z$ and $(w', v') \in R'_i$. Now applying the induction hypothesis, we get $v' \in W'_k$, and hence $(i, k) \in \sigma_{P'}(w')$, as required.

Claim 3: At the beginning of the Nth iteration, block W_i and W'_i contain the same signatures wrt. the current partitions. Proof: It suffices to prove that any signature occurring in W_i also occurs in W'_i (the other direction being symmetric). Consider any signature $\sigma_P(w)$ with $w \in W_i$. By Claim 1, there exists a $w' \in W'$ with $(w, w') \in Z$. By the induction hypothesis, $w' \in W'_i$. By Claim 2, $\sigma_{P'}(w') = \sigma_P(w)$, showing that $\sigma_P(w)$ also occurs in W'_i.

We need to show that if $(w, w') \in Z$, then w and w' end up in the same block after the Nth iteration of the repeat-until loop. Consulting lines 6–12 of the algorithm, it follows from Claim 3 that the Nth iteration will lead to the same splits for s and t and with the same signatures moved to the same new blocks. From Claim 2, we know that w and w' have the same signatures, and hence will be moved to the same new blocks (or stay where they are), as required. □

Theorem 1. Suppose $s \leftrightarrows t$. The contracted states returned by running $OrderedPartitionRefinement$ on s and t are identical.

Proof. Suppose $s = (W, R, L, W_d)$ and $s' = (W', R', L', W'_d)$ is the contraction of s (the output of running $OrderedPartitionRefinement$ on s). Similarly, suppose $t = (V, Q, K, V_d)$ and $t' = (V', Q', K', V'_d)$ is the contraction of t. By Claim 3 of the proof of Lemma 1, we know that the final partitions at the end of the repeat-until loop will have the same length and pairwise contain the same signatures for both runs of $OrderedPartitionRefinement$. Let $P = (W_1, \ldots, W_k)$ denote the final partition of s and $P' = (V_1, \ldots, V_k)$ the final partition of t. Both s' and t' will then have the same set of worlds $\{1, \ldots, k\}$. Since the blocks have pairwise identical signatures, each world m will have the same label in both s' and t'. Also, world m will have the same outgoing edges in both s' and t': $(m, n) \in R'_i \Leftrightarrow$ there exists $w, v \in W$ such that $(w, v) \in R_i$, $w \in W_m$ and $v \in W_n \Leftrightarrow (i, n)$ is an element of a signature occurring in W_m wrt. partition $P \Leftrightarrow (i, n)$ is an element of a signature occurring in V_m wrt. partition $P' \Leftrightarrow$ there exists $w', v' \in V$ such that $(w', v') \in Q_i$, $w' \in V_m$ and $v' \in V_n \Leftrightarrow (m, n) \in Q'_i$. Finally, suppose $m \in W'_d$. We then need to prove $m \in V'_d$ (the other direction being symmetric). Since $m \in W'_d$, block W_m must contain a world $w \in W_d$. Then since $s \leftrightarrows t$, by [designated] we get the existence of a $w' \in V_d$ with $(w, w') \in Z$. It then follows from Lemma 1 that w' must be in block V_m. Hence $V_m \cap V_d \neq \emptyset$, so $m \in V'_d$, as required. □

5 Planning Tasks and Policies

With epistemic states and actions defined, we can now describe how these concepts can be used for planning. We follow the approaches of Engesser et al. (2017) and Bolander et al. (2018), but describe a number of modifications and additions to make policies more intuitively sensible and to make the computation of policies more efficient.

Definition 5. A *planning task* for an agent $i \in \mathcal{A}$ is a tuple $\Pi = (s_0, A, \gamma)$, where s_0 is a local state for agent i called the *initial (epistemic) state*, A is a finite set of epistemic actions called the *action library* and γ is a common knowledge formula called the *goal formula*. For each $j \in \mathcal{A}$, we use A_j to denote the subset of A that consists only of the actions of agent j.

The reason we require goal formulas to be common knowledge formulas (formulas of the form $C\phi$) is that we are considering implicitly coordinated policies where no policy is agreed on between the agents in advance. If the goal formula is not a common knowledge formula, it could then happen that some agent keeps acting after another agent already ensured the goal to be satisfied. If we include public announcements in our action libraries, agents knowing to have reached the goal can publicly announce this and hence it will become common knowledge.

Example 3. We can now finalise the formalisation of the scenario described in the introduction, as the planning task

$\Pi = (s_0, A, \gamma)$ for r where s_0 is the state described in Example 1, $\gamma = C(K_h In(glasses, box_1) \vee K_h In(glasses, box_2))$ and A is the action library consisting of the $i{:}perceive(\phi)$ and $i{:}announce(\phi)$ actions from Figure 2 for $i = r, h$ and $\phi = In(glasses, box_1), In(glasses, box_2)$.

A solution to a planning task $\Pi = (s_0, A, \gamma)$ is a policy. Engesser et al. (2017) define two types of policies, *joint policies* and *global policies*. A *joint policy* is a collection of single-agent policies, one for each agent. Each single-agent policy maps local states of the relevant agent into actions. A *global policy* is a single multi-agent policy specifying in each global state the actions taken by any agent in that state. As shown by Engesser et al. (2017), the two types of policies are equivalent in terms of expressive power. In this paper, we have decided to work with joint policies rather than global policies. One of our motivations for using joint policies is that the policy definition becomes simpler, e.g. we don't need to add an extra condition to ensure uniformity.[1]

As earlier argued, we need our policies to be uniform, i.e., map indistinguishable states into the same actions. We also argued that the natural notion of indistinguishability on epistemic states is bisimilarity. As shown by Theorem 1, each class of bisimilar states has a unique representative that can be computed using $OrderedPartitionRefinement$. Hence by defining policies to be mappings from such unique representatives into actions, the uniformity condition will automatically be satisfied. We use S_i^{min} to denote the set of contracted (by $OrderedPartitionRefinement$) local states for agent i in which W_d is a minimal set closed under R_i.

Definition 6. Let $\Pi = (s_0, A, \gamma)$ be a planning task. A *(joint) policy* $\pi = (\pi_j)_{j \in \mathcal{A}}$ for Π consists of partial mappings $\pi_j{:}S_j^{min} \to A_j$ satisfying that whenever $\pi_j(s)$ is defined, it is an action applicable in s.

The definition above is a revised version of the one by Engesser et al. (2017). In our version we make sure that policies are uniform with respect to indistinguishability based on modal equivalence, which was not guaranteed by the definitions of Engesser et al. We now turn to define executions of policies and solutions to planning tasks. Bolander et al. (2018) already provided such definitions for global policies, but we need to adapt them to the case of joint policies and our new notion of indistinguishability. We use $\lfloor s \rfloor$ to denote the bisimulation contraction of s using $OrderedPartitionRefinement$.

Definition 7. Let $\Pi = (s_0, A, \gamma)$ be a planning task and $\pi = (\pi_j)_{j \in \mathcal{A}}$ a policy for Π. An *execution* of π is a maximal sequence $(s_0, (g_0, j_0{:}a_0), s_1, (g_1, j_1{:}a_1), \dots)$ satisfying, for all $m \geq 0$,

1. $g_m \in Globals(s_m)$ (the execution picks a global state g_m from the current local state s_m)
2. $\pi_{j_m}(\lfloor g_m^{j_m} \rfloor) = j_m{:}a_m$ (the execution picks an agent j_m to perform the next action, computes the contracted local state $\lfloor g_m^{j_m} \rfloor$ for that agent, and finds the action $j_m{:}a_m$ specified by the policy for that agent)
3. $s_{m+1} = \lfloor g_m^{j_m} \otimes j_m{:}a_m \rfloor$ (the next local state of the execution is the contracted result of executing the chosen action in the chosen state)

An execution is called *successful* if it is finite and its last element satisfies γ. The policy π is *implicitly coordinated* for Π if every execution is successful.

The policies are called implicitly coordinated since we not only require all objectively possible executions to be successful, but also all *subjectively possible executions*, i.e. the executions in which we always replace the current global state by its perspective shift to the next agent who will act (see conditions 2 and 3 above). This implies that policies will always be verified from the local perspective of the agent acting next. For a more thorough explanation and discussion of these issues, consult Engesser et al. (2017) and Bolander et al. (2018).

6 Algorithm

We will now present our planning algorithm for computing implicitly coordinated policies with optimal worst-case execution length. The algorithm is a Breadth-First Search (BFS) variant of the classical AND-OR search (Nilsson 1971) on a rooted graph. We use a modified version of PrAO (Pruning AND-OR search) first proposed by To et al. (2011) and further detailed by To (2012), with the same definitions for *solved* and *dead* nodes, but using explicit and-nodes and delaying the policy extraction until after the search. Distinct nodes in the graph will represent distinct non-bisimilar states (it is a graph search, not a tree search). Therefore state and node will be used interchangeably in the following. When expanding a node s, the algorithm applies the following node expansion rule: If s is an or-node, create a child and-node for every action $j{:}a \in A$ applicable in s^j. Otherwise, create a child or-node for every global state contained in s. When branching on an action $j{:}a$ from an or-node s, the algorithm calculates the product update $s' = s^j \otimes j{:}a$, and then applies ordered partition refinement to contract it, producing a unique minimal state for each set of bisimilar states.

The pseudocode of our planning algorithm is provided in Algorithm 2. Line 1 has been contracted due to space limitations. It performs some bookkeeping by creating a root node from s_0, and adding it to the graph as an and-node. It then essentially runs lines 10-17 where s' is replaced by the root. Thus the root becomes solved if all of its globals become solved. Lines 2-5 iterate the frontier starting at the globals of the root, s_0, and check for all actions $j{:}a \in A$ if they are applicable from the perspective of j. Since the search is breadth-first, we use a queue for the frontier. Lines 6-9 perform the perspective shift, product update and contraction. If the same contracted state exists anywhere in the graph, it will be set as a child of the current node s. Otherwise, a new node is created and added to the graph as a child of s. Note that to prevent a global state t from matching its child t', i.e. when $Globals(t) = \{t'\}$, we only compare nodes of

[1] Another original motivation was that we managed to prove that joint policies can be up to quadratically smaller than their corresponding global policies, potentially giving them a computational advantage. However, there might be ways to represent global policies more compactly, so it would require a deeper investigation to prove that joint policies are *necessarily* computationally advantageous.

Algorithm 2: *Plan*

Input: A planning task $\Pi = (s_0, A, \gamma)$ for agent $i \in \mathcal{A}$
Output: Policy π for Π, or *failure*

```
1  initialize(i, s_0)
2  while not frontier.empty() do
3      s ← frontier.dequeue()
4      for each j:a ∈ A do
5          if not s^j.applicable(j:a) then continue
6          s' ← s^j ⊗ j:a
7          s' ← OrderedPartitionRefinement(s')
8          if s' ∈ S_and then continue
9          S_and ← S_and ∪ {s'}
10         for each s'' ∈ Globals(s') do
11             s'' ← OrderedPartitionRefinement(s'')
12             if s'' ∈ S_or then continue
13             S_or ← S_or ∪ {s''}
14             if s'' ⊨ γ then
15                 update_solved_dead(s'')
16             else
17                 frontier.enqueue(s'')
18     update_solved_dead(s)
19     if solved(root) then return extract_policy()
20     if dead(root) then return failure
21 return failure
```

the same type (and-/or-nodes). Lines 10-17 iterate the global states of s' and repeat the previously described check for if the contracted state s'' already exists in the graph. Since s'' is an or-node, it is checked whether it fulfills the goal formula γ, in which case it is set to solved, and propagated, otherwise it is added to the frontier for further expansion.

Lines 15 and 18 check (after each found goal and completed node expansion, respectively) if a given node t is *solved*, *dead* or *undetermined* (status not yet known), updates a flag on the node accordingly, and propagates this up the graph in case the node is *solved* or *dead*. The propagation recursively updates the status of all parents if the status of the updated node changes (i.e., if a parent t' of t changes from undetermined to solved or dead, then the parents of t' are updated as well). The intuition is that a node is solved if it (for or-nodes) is a goal state or has an action which necessarily leads to a goal state, or (for and-nodes) all its global states necessarily lead to a goal state. A node is called *expanded* if all its children have been generated. Formally, $solved(t)$ and $dead(t)$ are recursively defined for expanded nodes by:

- And-node: $solved(t)$ is true iff all children of t are solved.
- Or-node: $solved(t)$ is true iff $t \vDash \gamma$ or t has a solved child.
- And-node: $dead(t)$ is true iff t has a dead child.
- Or-node: $dead(t)$ is true iff all children of t are dead and $t \nvDash \gamma$.

An *undetermined* node is one which is neither *solved* nor *dead*. Lines 19 and 20 check for termination. If $dead(root)$ then no policy exists, else if $solved(root)$ the policy is extracted and returned. The policy extraction performs two graph traversals, only visiting solved nodes. The first is a bottom-up iterative traversal, where we start by marking all solved leaf nodes with a cost of 0 and all other nodes with a special undefined cost. Then in the i'th iteration, we perform the following check for all parents of nodes with cost $i-1$: if the parent has an undefined cost and is an or-node, we mark it with cost i; if the parent is an and-node and all its children have defined costs, we mark it with the maximal cost of its children. We continue this until the root node is reached. The second traversal is top-down, starting at the root, visiting (for or-nodes) the child with the lowest cost, and (for and-nodes) all children. At an or-node with state t, and the action $j:a$ to the lowest cost child, the policy π_j is extended with the assignment $\lfloor t^j \rfloor \rightarrow j:a$. See Supplementary Material for a concrete example of the policy extraction. Note that even though the graph may contain cycles, the extracted policy doesn't. When traversing from the root to a leaf node, always choosing the lowest cost child at an or-node, the cost will always decrease by at least one per step. If two copies of the same state existed on a traversed path from root to leaf, the cost must have had a non-decreasing change at some step, which is contradictory.

Soundness and completeness of Algorithm 2 can be shown by adopting the standard soundness and completeness proofs for AND-OR search, cf. e.g. Theorem 10 of To (2012). The proof of Theorem 10 would have to be adopted to our slight variant of the algorithm (explicit and-nodes and policy extraction after ended search), the use of epistemic states and contractions, and the fact that our state spaces can be potentially infinite (Bolander and Andersen 2011) (hence termination for unsolvable planning tasks is not guaranteed in our case).

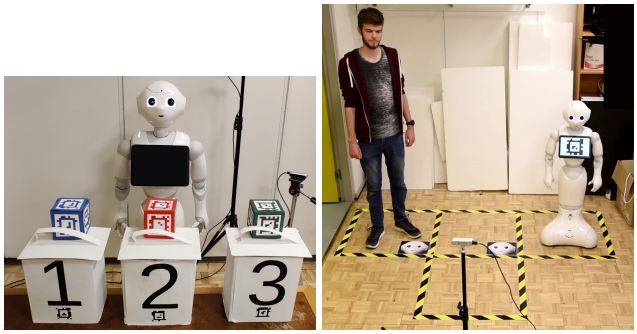

Figure 5: The Pepper robot used in our experiments. On the left we see part of the setup for the cubes and boxes domain. On the right we see a MAPF/DU problem instance where the barricade tape marks the grid.

7 Experiments and Results

We have implemented a planning system, using Algorithms 1 and 2, in the C++ programming language. The planner has been tested and evaluated using three domains which will be described in the following. We integrated the planner with a robot platform, and tested it on two of the domains involving human-robot collaboration. Source code for the planner and video of robot experiments is available at `https://github.com/Zeltex/Implicit_Coordination`.

Cubes and Boxes is an extended version of the domain described by Dissing and Bolander (2020). The domain contains two types of objects, cubes and boxes (See Figure 5 left). The cubes can be inside boxes, represented by the predicate $In(cube, box)$. The agents can perform the three types of generic actions described in Figure 2. In particular, we study a more generic version of the "*missing glasses*" scenario described in the introduction, where two agents, a robot r and a human h, are each looking for their missing cube. The setup consists of four boxes $box_1, \ldots, box_4$ and four cubes initially distributed into the four closed boxes. The robot wants to know the location of $cube_r$, the human the location of $cube_h$. The boxes are placed such that two are accessible to the human and the other two to the robot. The goal is $\gamma = C((\vee_{i=1}^{4} K_r In(cube_r, box_i)) \wedge (\vee_{j=1}^{4} K_h In(cube_h, box_j)))$. When applying the planner to this problem with the robot as the planning agent, the planner finds a policy in which the robot looks into each of its accessible boxes and announces if it has found the human's cube. Note that the speech act of announcing a cube location is treated as any other atomic action by the planner, since our planner is epistemic and therefore naturally able to incorporate information sharing in the planning process (note that the robot policy only informs the human about $cube_h$; if it discovers any other cube, it will not say anything).

Multi-Agent Pathfinding with Destination Uncertainty Task (MAPF/DU) was introduced by Bolander et al. (2018) and explored further by Nebel et al. (2019). Here a set of agents must collaboratively manoeuvre through a grid world (or, more generally, a graph) with the joint goal of each reaching its assigned destination. E.g. in the instance shown in Figure 5 right, the goal is for the human to get to the grid cell to the far right, and the robot the cell at the top. The problem is complicated by each grid cell only fitting a single agent and each agent only knowing its own destination (each agent only knows that the destination of the other agent is one among a set of *possible destinations*). The agents can't communicate, and thus need implicit coordination to ensure they both reach their final destinations. See Supplementary Material for a full formalisation of this example.

Coin flip is a simple demonstration of how bisimulation contraction can significantly reduce the amount of memory required. In the domain, two agents take turns flipping a coin, where the outcome of the coin flip is private to the acting agent. In brief, this causes the number of worlds in each state to grow exponentially in the number of actions performed, however since at any point in time the current outcome is either heads or tails, it is possible to contract the state down to two worlds. A detailed description of the domain can be found in the Supplementary Material.

Benchmarking All experiments were performed on a machine with dual Intel Xeon Gold 6126 CPUs, 256 GB of RAM reserved and a 48 hours timeout. We compare our planner with: 1) a baseline version which skips the partition refinement on lines 7 and 11 of Algorithm 2; and 2) the planner developed by Engesser et al. (2017). The main differences between our planner and the Engesser planner is: 1) They use comparison-based logarithmic-time state lookups while our unique state representatives allows constant-time lookups using hashing; 2) They use a dense bitset (1 bit per possible proposition) to represent sets of propositions where we use a sparse set of proposition ids; 3) They use simplified, partial contractions that, unlike ours, don't guarantee all bisimilar states to be identified.

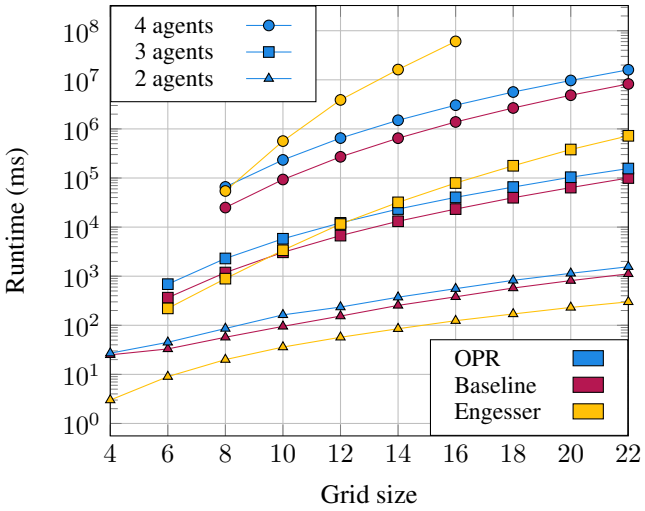

Figure 6: MAPF/DU benchmark comparing our planner without partition refinement (Baseline), our planner with ordered partition refinement (OPR), and the planner developed in Engesser et al. (2017). MAPF/DU instances are parameterised by grid size (total number of positions the agents can be in) and number of agents. Our planner with ordered partition refinement finishes the largest problem 19 times faster than the Engesser planner. The planner by Engesser timed out on the three largest problems.

The MAPF/DU results are shown in Figure 6. We use the same problem instances as Engesser and Miller (2020) (grid sizes 4-14) as well as larger instances following the same corridor pattern (grid sizes 16-22). The planner by Engesser is initially faster, but is overtaken by the baseline planner when reaching instances with $\geq$ 3 agents and $\geq$ 10 grid cells. This is partly due to the state hashing resulting in some overhead for the small problems, but much better scaling for the large problems, completing the instance with 4 agents and 16 grid cells around 43 times faster than the Engesser planner. All actions in this domain have a single event which results in no two states being bisimilar without also being identical. This means that the bisimulation contraction had no effect on the amount of hashes generated, and it therefore was strictly worse than the baseline due to the additional computation needed for ordered partition refinement. However, the numbers in the figure indicate that ordered partition refinement only adds a constant factor overhead.

The results for the coin flip domain can be found in Figure 7. Ordered partition refinement contracts the generated state such that it has exactly 2 worlds at all plan depths, making it scale more or less linearly with the plan depth. The baseline, and the simplified contraction in the planner by Engesser is not able to contract the state which doubles in size

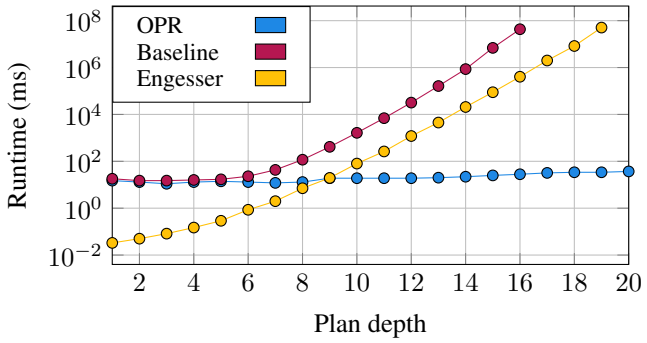

Figure 7: Coin flip benchmark comparing our planner (OPR) with the baseline and the planner of Engesser et al. (2017).

Algorithm 3: *AgentLoop* for the robotic agent i

Input: A planning task $\Pi = (s_0, A, \gamma)$ for agent $i \in \mathcal{A}$

$s \leftarrow s_0$
$\pi \leftarrow Plan(\Pi)$ `/* call Algorithm 2 */`
while $s \not\models \gamma \wedge \pi \neq$ *failure* **do**
  for each $j{:}a, \rho$ *in* $perceived\_actions()$ **do**
    $s \leftarrow perception\_update(s \otimes j{:}a, \rho)$
  switch s **do**
    case $s \in Dom(\pi_i)$ **do** $execute(\pi_i(s))$
    case $\forall g \in Globals(s), \exists j \in \mathcal{A}$ s.t. $\lfloor g^j \rfloor \in Dom(\pi_j)$ **do continue**
    otherwise do $\pi \leftarrow Plan((s, A, \gamma))$
if $s \models \gamma$ **then return** *success* **else return** *failure*

(amount of worlds) after each action, making the planning time scale at least exponentially with the plan depth. It can be seen that our planner has some overhead due to ordered partition refinement, but it is quickly outweighed by the exponential scaling of the baseline and planner by Engesser.

In conclusion we see that in domains that generate large contractable states like the coin flip domain, ordered partition refinement is highly useful. In contrast, in domains which do not benefit from bisimulation contraction like MAPF/DU, ordered partition refinement adds a manageable constant factor overhead.

Robotic Integration The planner has furthermore been integrated with an extended version of the robotic platform described by Dissing and Bolander (2020), and used to solve robotic variants of "Cubes and Boxes" and MAPF/DU with two agents. In brief, video from two Intel RealSense RGB+D cameras is sent to a number of *detectors*, each of which recognise some specific feature such as faces, object markers and body poses. Each detector produces a stream of *percepts* containing the recognised features and their spatial positions. These percepts are linked to agents and objects in the internal world model through an anchoring process (Coradeschi and Saffiotti 2000; Williams et al. 2009). Actions performed by the observed agents are detected by continuously monitoring for changes in the world model. The perception pipeline interfaces with the planner through the perceive-plan-act agent loop described in Algorithm 3.

Given a planning task the agent loop first computes an initial policy and repeats the following procedure until either a goal state is reached or no successful policy can be found: First, update the current epistemic state by applying all actions detected by the perception pipeline since the last iteration (including those performed by the robot itself). Next, if the policy defines an action for the robot in this new state, it is immediately executed. Otherwise, if in all global states, there is some agent from whose perspective the policy defines an action, the robot just continues observing, waiting for some other agent to act. Finally, if neither condition holds for the current state, e.g. due to some agent having performed an unexpected action, the agent loop replans from this unexpected state. A limitation of the formalism is the assumption of sequential action execution, i.e., when an agent performs an action, all other agents must remain stationary for the duration of the action in order to ensure that the percepts are generated in the correct order. We use the LEDs on the robot to communicate this to the human agent, e.g. green lights means it is okay for the human to move. Extending the formalism to concurrent actions is important future work.

Updating a state s based on an observed action $j{:}a$ consists of two steps: 1) the product update $s \otimes j{:}a$ predicts the possible outcomes of executing the action in s and 2) the *perception_update* function filters out all outcomes not consistent with actual observations (the action $j{:}a$ is not necessarily among the actions in the action library A of the planning problem: It is the local perspective of agent i on the action executed by j). More precisely, the perception pipeline produces a *percept* formula ρ of what was actually observed by the robot during the execution of an action and *perception_update* then produces the correct local state by removing all designated worlds from $s \otimes j{:}a$ which do not satisfy ρ. E.g. suppose the robot performed $r{:}perceive(In(glasses, box_1))$ in s_0 from Figure 1. First the agent loop predicts the possible outcomes by computing the product update resulting in state s_1 from Figure 3, with two outcomes distinguishable by the robot. While the robot executes the physical action, the perception system either produces $In(glasses, box_1)$ or $\neg In(glasses, box_1)$ based upon the observation, which then allows *perception_update* to correctly select the actual outcome.

8 Current Limitations

A significant limitation of our current approach is the restriction to reasoning with **S5** logic and thereby the inability to reason about the (possibly incorrect) beliefs of other agents. A second limitation is that the current algorithm is not yet capable of finding cyclic policies, which might be required in some planning tasks with non-deterministic actions. A third limitation is in the explicit representation of uncertainty as the set of all possible worlds, where more compact representations might scale significantly better, e.g. using symbolic knowledge structures as presented by van Benthem et al. (2018) with perspective shifts as recently developed by Gattinger (2020).

References

Aceto, L.; Ingólfsdóttir, A.; and Srba, J. 2012. The algorithmics of bisimilarity. In *Advanced Topics in Bisimulation and Coinduction*, volume 52 of *Cambridge tracts in theoretical computer science*. Cambridge, England: Cambridge University Press. 100–172.

Baltag, A.; Moss, L. S.; and Solecki, S. 1998. The logic of public announcements and common knowledge and private suspicions. In *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98)*, 43–56. Evanston, IL, USA: Morgan Kaufmann.

Blackburn, P., and van Benthem, J. 2007. Modal logic: A semantic perspective. In *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*. North-Holland. 1–84.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge, UK: Cambridge University Press.

Bolander, T., and Andersen, M. B. 2011. Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics* 21:9–34.

Bolander, T.; Engesser, T.; Mattmüller, R.; and Nebel, B. 2018. Better eager than lazy? How agent types impact the successfulness of implicit coordination. In *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*. Tempe, Arizona, USA: AAAI Press.

Bolander, T. 2017. A gentle introduction to epistemic planning: The DEL approach. In *Proceedings of the Ninth Workshop on Methods for Modalities*, volume 243 of *EPTCS*, 1–22.

Buisan, G., and Alami, R. 2021. A human-aware task planner explicitly reasoning about human and robot decision, action and reaction. In Bethel, C. L.; Paiva, A.; Broadbent, E.; Feil-Seifer, D.; and Szafir, D., eds., *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction, HRI 2021, Boulder, CO, USA, March 8-11, 2021*, 544–548. ACM.

Coradeschi, S., and Saffiotti, A. 2000. Anchoring symbols to sensor data: Preliminary report. In Kautz, H. A., and Porter, B. W., eds., *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA*, 129–135. AAAI Press / The MIT Press.

Dissing, L., and Bolander, T. 2020. Implementing theory of mind on a robot using Dynamic Epistemic Logic. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 1615–1621. Yokohama, Japan: IJCAI.

Engesser, T., and Miller, T. 2020. Implicit coordination using FOND planning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 7151–7159. New York, NY, USA: AAAI Press.

Engesser, T.; Bolander, T.; Mattmüller, R.; and Nebel, B. 2017. Cooperative epistemic multi-agent planning for implicit coordination. In *Proceedings of Methods for Modalities*, volume 243 of *Electronic Proceedings in Theoretical Computer Science*, 75–90.

Gattinger, M. 2020. Towards symbolic and succinct perspective shifts. *Epistemic Planning workshop at the 30th International Conference on Automated Planning and Scheduling*.

Jamroga, W., and Aagotnes, T. 2007. Constructive knowledge: What agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics* 17(4):423–475.

Lemaignan, S., and Dillenbourg, P. 2015. Mutual modelling in robotics: Inspirations for the next steps. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 303–310. IEEE.

Nebel, B.; Bolander, T.; Engesser, T.; and Mattmüller, R. 2019. Implicitly coordinated multi-agent path finding under destination uncertainty: Success guarantees and computational complexity. *Journal of Artificial Intelligence Research* 64:497–527.

Nilsson, N. J. 1971. *Problem-solving Methods in Artificial Intelligence*. McGraw-Hill computer science series. New York, NY, USA: McGraw-Hill.

Reifsteck, D.; Engesser, T.; Mattmüller, R.; and Nebel, B. 2019. Epistemic multi-agent planning using Monte-Carlo Tree Search. In *KI 2019: Advances in Artificial Intelligence - 42nd German Conference on AI*, 277–289. Kassel, Germany: Springer.

Talamadupula, K.; Briggs, G.; Chakraborti, T.; Scheutz, M.; and Kambhampati, S. 2014. Coordination in human-robot teams using mental modeling and plan recognition. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, 2957–2962. IEEE.

Thomaz, A.; Hoffman, G.; and Cakmak, M. 2016. Computational human-robot interaction. *Foundations and Trends in Robotics* 4(2-3):105–223.

To, S.; Son, T.; and Pontelli, E. 2011. Contingent planning as and/or forward search with disjunctive representation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 21.

To, S. T. 2012. A new approach to contingent planning using a disjunctive representation in and/or forward search with novel pruning techniques. *Journal of Artificial Intelligence Research*.

van Benthem, J.; van Eijck, J.; Gattinger, M.; and Su, K. 2018. Symbolic model checking for dynamic epistemic logic - S5 and beyond. *Journal of Logic and Computation* 28(2):367–402.

Williams, M.-A.; McCarthy, J.; Gärdenfors, P.; Stanton, C. J.; and Karol, A. 2009. A grounding framework. *Auton. Agents Multi Agent Syst.* 19(3):272–296.