A Gentle Introduction to Epistemic Planning: The DEL Approach

Thomas Bolander
DTU Compute
Technical University of Denmark
Copenhagen, Denmark
tobo@dtu.dk

Epistemic planning can be used for decision making in multi-agent situations with distributed knowledge and capabilities. Dynamic Epistemic Logic (DEL) has been shown to provide a very natural and expressive framework for epistemic planning. In this paper, we aim to give an accessible introduction to DEL-based epistemic planning. The paper starts with the most classical framework for planning, STRIPS, and then moves towards epistemic planning in a number of smaller steps, where each step is motivated by the need to be able to model more complex planning scenarios.

1 Introduction

Automated planning is a branch of artificial intelligence concerned with computing plans (sequences of actions) leading to some desired goal. A human or robot could e.g. have the goal of picking up a parcel at the post office, and then the problem becomes to find a succesful sequence of actions achieving this. Epistemic planning is the enrichment of planning with epistemic notions, that is, knowledge and beliefs. The human or robot might have to reason about epistemic aspects such as: Do I know at which post office the parcel is? If not, who would be relevant to ask? Maybe the parcel is a birthday present for my daughter, and I want to ensure that she doesnt get to know, and have to plan my actions accordingly (make sure she doesn't see me with the parcel). The epistemic notions are usually formalised using an epistemic logic. Epistemic planning can naturally be seen as the combination of automated planning with epistemic logic, relying on ideas, concepts and solutions from both areas.

In general, epistemic planning considers the following problem: Given my current state of knowledge, and a desirable state of knowledge, how do I get from one to the other? It is of central importance in settings where agents need to be able to reason about their own lack of knowledge, and e.g. make plans of how to achieve the required knowledge. It is also essential in multi-agent planning, where succesful coordination and collaboration can only be expected if agents are able to reason about the knowledge, uncertainty and capabilities of the other agents.

In this gentle introduction to epistemic planning, the focus will be on the DEL approach: Using Dynamic Epistemic Logic (DEL) as the underlying formalism. We start with the most classical framework for planning, STRIPS, and then stepwise we expand and generalise the framework until finally reaching full multi-agent planning based on dynamic epistemic logic. Each of these steps will be based on the need to be able to formalise specific planning scenarios.

In Section 2 we will first introduce classical planning domains and planning tasks. Then we move to present the basics of the STRIPS planning framework in Section 3, and propositional planning in Section 4. We then slowly progress towards defining the epistemic planning framework via first defining belief states and conditional actions in Section 5, and then epistemic logic and dynamic epistemic logic in Sections 6–7. In Section 8 we finally define epistemic planning tasks, and study various extensions

Sujata Ghosh and R. Ramanujam: M4M9
EPTCS 243, 2017, pp. 1–22, doi:10.4204/EPTCS.243.1

and generalisations in Sections 9–11. In Section 12 we very briefly study complexity issues of epistemic planning, and we round off with a discussion of alternative approaches to epistemic planning in Section 13.

2 Classical planning domains and planning tasks

Example 1. Suppose a father has his daughter's birthday coming up, and he ordered a present for her which is now at the post office. His goal is to be able to give her the present the next day. This is a **planning task** (sometimes called a **planning problem**): He needs to compute a plan to achieve the goal. In a planning task, one is given an **initial state**, a set of **goal states** and a set of available **actions**. The problem is now to compute a sequence of actions (a **plan**) that, if executed in the initial state, will lead to one of the goal states. In the birthday present example, the initial state describes that the present is at the post office and not yet wrapped. The goal states are those in which the present is at home and wrapped (ready to be given on the next day). The available actions could be actions like going from home to the post office, going from the post office to home, picking up the present at the post office, and wrapping the present. Of course we do not need to limit ourselves to only allowing these specific actions, but could have general actions for going from a location A to a location B, general actions for picking up an object at a location, and general actions for wrapping an object that you are currently holding.

To allow us to reason formally about planning tasks and plans, and to allow computers and robots to compute plans, we need an appropriate formalism to describe such objects. The simplest approach is to define planning tasks in terms of state-transition systems.

Definition 1. [17] A **(restricted) state-transition system** (also called a **classical planning domain** or simply a **state space**) is $\Sigma = (S, A, \gamma)$ where:

- S is a finite or recursively enumerable set of **states**.
- A is a finite or recursively enumerable set of **actions**.
- $\gamma : S \times A \hookrightarrow S$ is a computable partial **state-transition function**.

When $\gamma(s,a)$ is defined, a is said to be **applicable** in s. When $\pi = a_1; \cdots ; a_n$ is a sequence of actions from A, we write $\gamma(s,\pi)$ for $\gamma(\ldots\gamma(\gamma(\gamma(s,a_1),a_2),a_3),\ldots,a_n)$.[1]

Definition 2. [17] A **classical planning task** is a triple (Σ, s_0, S_g) where:

- $\Sigma = (S, A, \gamma)$ is a state-transition system (a classical planning domain).
- $s_0 \in S$ is the **initial state**.
- $S_g \subseteq S$ is the set of **goal states**.

A **solution** to a classical planning task $((S, A, \gamma), s_0, S_g)$ is a finite sequence of actions (a **plan**) $\pi = a_1; \cdots ; a_n$ from A such that $\gamma(s_0, \pi) \in S_g$. The **length** of a solution π is the number of actions in π.

Example 2. Consider the birthday present example from Example 1. The available actions and their corresponding state transitions could be presented by the state-transitions system of Figure 1. Here we have $\Sigma = (S, A, \gamma)$ with $S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$, $A = \{$go to post office, go home, pick up present, wrap present$\}$, and γ is as given in the figure (e.g. $\gamma(s_1, \text{go to post office}) = s_2$ since there is an edge labelled "go to post office" from s_1 to s_2). Note that "wrap present" is only applicable after "pick up present" has been executed: it is necessary to get hold of the present before it can be wrapped. The

[1] So $\gamma(s,\pi)$ is the result of executing the actions of π in sequence starting in s.

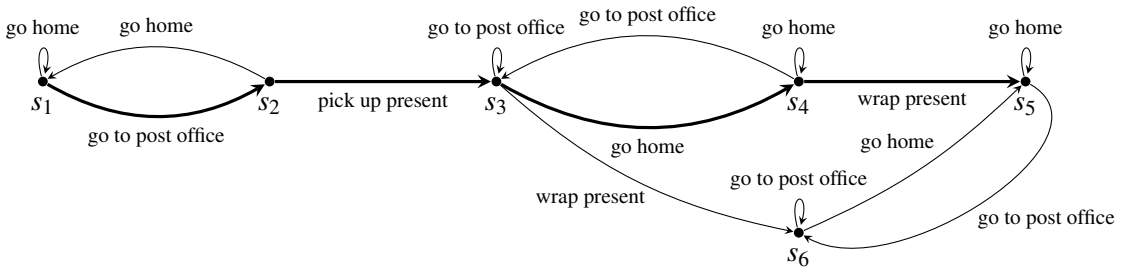


Figure 1: A state-transition system for the birthday present example.

planning task of Example 1 can then be represented as the classical planning task (Σ, s_0, S_g) where $s_0 = s_1$ and $S_g = \{s_5\}$. A solution is highlighted in Figure 1:

$$\pi = \text{go to post office; pick up present; go home; wrap present.}$$

Representing planning tasks directly as state-transition systems has some important weaknesses: 1) there is no internal structure on states and actions to understand or clarify what they represent; 2) state-transition systems are normally of exponential size in the number of propositional variables required to describe the states. To exemplify the second weakness, if the planning task was to take home n parcels from the post office, the state space (state-transition system) would be of size $\geq 2^n$: each parcel could either be at the post office or at home. This is so even though the length of the shortest solution would only be linear in n (e.g. bring the parcels home one at a time). To address both of these weaknesses one introduces logical structures on states and actions. This is e.g. done in the planning language STRIPS to be introduced next.

3 STRIPS planning

The classical language for describing states and actions in the field of automated planning is STRIPS [16]. We will here introduce STRIPS slightly informally, and the interested reader is referred to [29] for more details. The reader familiar with STRIPS planning can skip this section, but might want to briefly look at the examples.

In STRIPS, states are represented as sets of ground atoms of a function-free first-order language $\mathcal{L}$. In the birthday present example, the initial state could e.g. be described by

$$\begin{aligned} s_0 = \quad & \{\mathsf{At}(\mathsf{Father},\mathsf{Home}), \mathsf{At}(\mathsf{Present},\mathsf{PostOffice}), \mathsf{IsAgent}(\mathsf{Father}), \mathsf{IsLocation}(\mathsf{Home}), \\ & \mathsf{IsLocation}(\mathsf{PostOffice}), \mathsf{IsObject}(\mathsf{Present})\} \end{aligned} \tag{1}$$

where $\mathsf{At}(x,y)$ is a predicate for expressing that object (or agent) x is at location y. The predicates $\mathsf{IsAgent}(x)$, $\mathsf{IsLocation}(x)$ and $\mathsf{IsObject}(x)$ are true of agents, objects and locations, respectively (alternatively, one could use a sorted first-order language and then omit these predicates). Actions are described via so-called **action schemas**. The actions of *going from one location to another*, *picking up an object* and *wrapping an object* can be expressed by the STRIPS action schemas provided in Figure 2, where predicates have names starting with upper-case letters, and variables have names starting with lower-case letters. Each action schema has a **name**, a **precondition** and an **effect**. Preconditions and effects

ACTION : $\mathsf{Go}(agt, from, to)$
PRECOND : $\mathsf{At}(agt, from) \wedge \mathsf{IsAgent}(agt) \wedge \mathsf{IsLocation}(from) \wedge \mathsf{IsLocation}(to)$
EFFECT : $\mathsf{At}(agt, to) \wedge \neg\mathsf{At}(agt, from)$

ACTION : $\mathsf{PickUp}(agt, obj, from)$
PRECOND : $\mathsf{At}(agt, from) \wedge \mathsf{At}(obj, from) \wedge \neg\mathsf{Has}(agt, obj) \wedge \mathsf{IsAgent}(agt) \wedge \mathsf{IsObject}(obj) \wedge \mathsf{IsLocation}(from)$
EFFECT : $\mathsf{Has}(agt, obj) \wedge \neg\mathsf{At}(obj, from)$

ACTION : $\mathsf{Wrap}(agt, obj)$
PRECOND : $\mathsf{Has}(agt, obj) \wedge \neg\mathsf{Wrapped}(obj) \wedge \mathsf{IsAgent}(agt) \wedge \mathsf{IsObject}(obj)$
EFFECT : $\mathsf{Wrapped}(obj)$

Figure 2: The set of STRIPS action schemas for the birthday present example.

are conjunctions of literals of the first-order language $\mathcal{L}$. A **ground action** is achieved by instantiating all variables of an action schema with constants of $\mathcal{L}$. For instance,

ACTION : $\mathsf{Go}(\mathsf{Father}, \mathsf{Home}, \mathsf{PostOffice})$
PRECOND : $\mathsf{At}(\mathsf{Father}, \mathsf{Home}) \wedge \mathsf{IsAgent}(\mathsf{Father}) \wedge \mathsf{IsLocation}(\mathsf{Home}) \wedge \mathsf{IsLocation}(\mathsf{PostOffice})$
EFFECT : $\mathsf{At}(\mathsf{Father}, \mathsf{PostOffice}) \wedge \neg\mathsf{At}(\mathsf{Father}, \mathsf{Home})$

The precondition of a ground action describes what has to be true for the action to be applicable.[2] It is easy to check that $\mathsf{Go}(\mathsf{Father}, \mathsf{Home}, \mathsf{PostOffice})$ is applicable in the initial state s_0 defined by (1) (all precondition atoms of the action occur in s_0). The effect of a ground action describes how the state is modified when the action is executed. The effect of $\mathsf{Go}(\mathsf{Father}, \mathsf{Home}, \mathsf{PostOffice})$ expresses that $\mathsf{At}(\mathsf{Father}, \mathsf{PostOffice})$ becomes true, and that $\mathsf{At}(\mathsf{Father}, \mathsf{Home})$ becomes false. Hence the result of executing $\mathsf{Go}(\mathsf{Father}, \mathsf{Home}, \mathsf{PostOffice})$ in s_0 will be the state

$$s_1 = \{\mathsf{At}(\mathsf{Father}, \mathsf{PostOffice}), \mathsf{At}(\mathsf{Present}, \mathsf{PostOffice}), \mathsf{IsAgent}(\mathsf{Father}), \mathsf{IsLocation}(\mathsf{Home}), \mathsf{IsLocation}(\mathsf{PostOffice}), \mathsf{IsObject}(\mathsf{Present})\}$$

Any finite set of STRIPS action schemas A **induce** a state-transition system (classical planning domain) $\Sigma = (S, A', \gamma)$ by:

- $S = 2^P$, where P is the set of ground atoms of $\mathcal{L}$.
- $A' = \{\text{all ground instances of the action schemas in } A\}$
- $\gamma(s,a) = \begin{cases} (s - \{\varphi \mid \neg\varphi \text{ is a negative literal of } \textsc{Effect}(a)) \cup \\ \quad \{\varphi \mid \varphi \text{ is a positive literal of } \textsc{Effect}(a)\} & \text{if } s \models \textsc{Precond}(a) \\ \text{undefined} & \text{otherwise} \end{cases}$

The state-transition system induced by the STRIPS schemas of Figure 2 is provided in Figure 3. The **rigid atoms**, those that cannot change truth-value, have been omitted. Furthermore, each constant name is abbreviated by the capital letters it contains (so for instance PostOffice is abbreviated PO). Note that

[2] More formally, a ground action a is applicable in a state s if $s \models \textsc{Precond}(a)$ where $\models$ denotes semantical entailment on propositional logic over the set of ground atoms of $\mathcal{L}$.

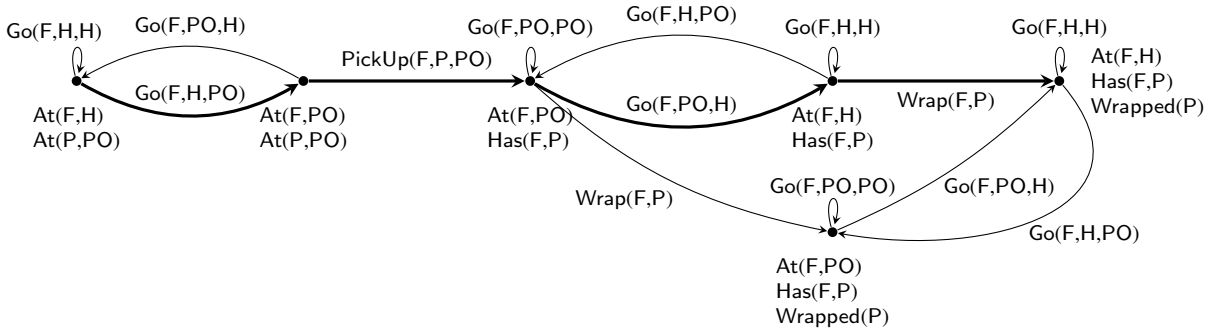


Figure 3: The state-transition system induced by the action schemas of Figure 2.

the resulting state-transition system is isomorphic to the state-transition system of Figure 1.

The advantage of the current STRIPS representation over the previous pure state-transition representation is that there is now structure on states and actions. This for instance means that the current formulation is easily generalisable, e.g. we can add $n-1$ additional parcels to the initial state without any need to modify the underlying action schemas. The induced state-transition system would become exponentially bigger, as earlier noted, but the size of the STRIPS action schemas would stay the same (though the size of the description of the initial state would grow linearly with n).

A **STRIPS planning task** on a function-free first-order language $\mathcal{L}$ is (A, s_0, φ_g) where A is a set of STRIPS action schemas over $\mathcal{L}$, s_0 is a set of ground atoms over $\mathcal{L}$ and φ_g is a conjunction of ground literals over $\mathcal{L}$ called the **goal formula**. Any STRIPS planning task (A, s_0, φ_g) induces a classical planning task (Σ, s_0, S_g) by letting Σ be the state-transition system induced by A and letting $S_g = \{s \in S \mid s \models \varphi_g\}$. A **solution** to a STRIPS planning task is then a solution to the induced classical planning task.

Example 3. The birthday present example can be represented as a STRIPS planning task (A, s_0, φ_g) where A is the set of action schemas of Figure 2, s_0 is defined by (1), and $\varphi_g = \mathsf{At}(\mathsf{Father}, \mathsf{Home}) \wedge \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) \wedge \mathsf{Wrapped}(\mathsf{Present})$. This is a STRIPS planning task on the function-free first order language $\mathcal{L}$ with binary predicate symbols At and Has, unary predicate symbols Wrapped, IsAgent, IsLocation and IsObject, and constant symbols Home, PostOffice, Father, Present. Consulting Figure 3, it is easy to show that a solution to this planning task is

$$\begin{aligned}\pi = \quad & \mathsf{Go}(\mathsf{Father}, \mathsf{Home}, \mathsf{PostOffice}); \mathsf{PickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PostOffice}); \\ & \mathsf{Go}(\mathsf{Father}, \mathsf{PostOffice}, \mathsf{Home}); \mathsf{Wrap}(\mathsf{Father}, \mathsf{Present}).\end{aligned}$$

In the field of automated planning, actions are always described compactly in an action description language like STRIPS (or e.g. PDDL, ADL or SAS$^+$). A lot of research effort goes into finding ways to automatically derive efficient heuristics from action schemas, such that solutions can be found with minimal search. If given an induced state-transition system of a set of STRIPS action schemas, finding a solution to a planning task becomes a simple graph search problem (find a path from s_0 to a state in S_g). This can be done in linear time in the size of the state-transition system. However, as earlier noted, the induced state-transition system is often exponential in the size of the action schemas. Hence the complexity of computing solutions or deciding whether a solution exists (the **plan existence problem**) is in automated planning always measured in the size of the compact action schema representation. This is different from many formalisms in logic that consider plans or strategies and where complexity is measured in terms of the size of the state space. This is e.g. why epistemic planning based on ATEL

in [19] can be claimed to be tractable, even though already basic propositional STRIPS planning, which is much less expressive, is intractable. For most planning domains considered in automated planning (e.g. the planning domains of the International Planning Competition, IPC), calculating the entire state-transition system is computationally infeasible, and the goal is then for the heuristics to be sufficiently efficient that only the most relevant parts of the state-transition system are explored.

4 Propositional planning tasks

Even though STRIPS action schemas are formulated using first-order logic, for most purposes we can consider STRIPS as a planning formalism based on propositional logic. To see this, we first define *propositional planning tasks*.

Definition 3. A **propositional planning task**[3] on a finite set of atomic propositions P is (A, s_0, φ_g) where

- A is a finite set of **actions**. Each action is a pair $a = \langle pre(a), post(a)\rangle$ where $pre(a)$ and $post(a)$ are conjunctions of propositional literals over P. The element $pre(a)$ is called the **precondition** of a and $post(a)$ its **postcondition**.
- s_0 is the **initial state**, a propositional state over P (a subset of P).[4]
- φ_g is the **goal formula**, a propositional formula over P.

A propositional planning task (A, s_0, φ_g) on P **induces** a classical planning task $((S, A, \gamma), s_0, S_g)$ in the expected way (compare with the classical planning task induced by a STRIPS planning task defined in Section 3):

- $S = 2^P$
- $\gamma(s,a) = \begin{cases} (s - \{p \mid \neg p \text{ is a negative literal of } post(a)) \cup \\ \quad \{p \mid p \text{ is a positive literal of } post(a)\} & \text{if } s \models pre(a) \\ \text{undefined} & \text{otherwise} \end{cases}$
- $S_g = \{s \in S \mid s \models \varphi_g\}$.

A **solution** to a propositional planning task is any solution to the induced classical planning task.

For any function-free first-order language $\mathcal{L}$, let $P_\mathcal{L}$ denote the set of ground atoms of $\mathcal{L}$. The set $P_\mathcal{L}$ can be thought of as the atomic propositions of a propositional language. Any quantifier-free ground formula of $\mathcal{L}$ is then at the same time a formula of propositional logic over $P_\mathcal{L}$. Any STRIPS planning task (A, s_0, φ_g) on $\mathcal{L}$ **induces** a propositional planning task (A', s_0, φ_g) on $P_\mathcal{L}$ by simply letting

$$A' = \{\langle \mathsf{Precond}(a), \mathsf{Effect}(a)\rangle \mid a \text{ is a ground instance of an action schema in } A\}.$$

It is easy to show that the STRIPS planning task (A, s_0, φ_g) and its induced propositional planning task (A', s_0, φ_g) both induce the same classical planning task. They thus also have the same solutions. Note that the conventions for preconditions and effects are a bit different in propositional planning tasks than in STRIPS planning tasks. We now write precondition and effect pairs of an action a as a pair of the form $\langle pre(a), post(a)\rangle$, where we have also relabelled effects as *postconditions*. The point of both these changes is to gradually move away from the classical conventions of STRIPS planning into the conventions of dynamic epistemic logic that we will later present a planning framework based on.

[3] Called a **set-theoretic planning task** in [17].

[4] In general, we will identify propositional states with subsets of P. A subset $S \subseteq P$ represents the propositional state s in which the elements of S are the atomic propositions true in s.

Example 4. The birthday present example can be represented as the propositional planning task (A, s_0, φ_g) below. It is a simplified version of the propositional planning task induced by the STRIPS planning task of Example 3, where we have done away with the rigid atoms that are no longer necessary. In the definitions below, *Agt* is a set of agent names (including Father), *Loc* is a set of locations (including Home and PostOffice) and *Obj* is a set of objects (including Present).

- $A = \{\mathsf{Go}(agt,from,to) \mid agt \in Agt \,\&\, from,to \in Loc\} \cup \{\mathsf{PickUp}(agt,obj,from) \mid agt \in Agt \,\&\, obj \in Obj \,\&\, from \in Loc\} \cup \{\mathsf{Wrap}(agt,obj) \mid agt \in Agt \,\&\, obj \in Obj\}$ where, for all $agt \in Agt$, all $from,to \in Loc$ and all $obj \in Obj$,
  - $\mathsf{Go}(agt,from,to) = \langle \mathsf{At}(agt,from), \mathsf{At}(agt,to) \wedge \neg \mathsf{At}(agt,from) \rangle$
  - $\mathsf{PickUp}(agt,obj,from) = \langle \mathsf{At}(agt,from) \wedge \mathsf{At}(obj,from) \wedge \neg \mathsf{Has}(agt,obj), \mathsf{Has}(agt,obj) \rangle$
  - $\mathsf{Wrap}(agt,obj) = \langle \mathsf{Has}(agt,obj) \wedge \neg \mathsf{Wrapped}(obj), \mathsf{Wrapped}(obj) \rangle$.
- $s_0 = \{\mathsf{At}(\mathsf{Father},\mathsf{Home}), \mathsf{At}(\mathsf{Present},\mathsf{PostOffice})\}$.
- $\varphi_g = \mathsf{At}(\mathsf{Father},\mathsf{Home}) \wedge \mathsf{Has}(\mathsf{Father},\mathsf{Present}) \wedge \mathsf{Wrapped}(\mathsf{Present})$.

Note that expressions like $\mathsf{At}(agt,from)$ with $agt \in Agt$ and $from \in Loc$ are no longer considered as ground atoms of the original first-order language $\mathcal{L}$, but as atoms of propositional logic over $P_{\mathcal{L}}$. A solution to this planning task is exactly as to the original STRIPS version: $\pi = \mathsf{Go}(\mathsf{Father},\mathsf{Home},\mathsf{PostOffice})$; $\mathsf{PickUp}(\mathsf{Father},\mathsf{Present},\mathsf{PostOffice})$; $\mathsf{Go}(\mathsf{Father},\mathsf{PostOffice},\mathsf{Home})$; $\mathsf{Wrap}(\mathsf{Father},\mathsf{Present})$.

Since any STRIPS planning task can be *propositionalised* as above, it means we can now work in a simpler formalism, propositional logic, which also makes it easier to generalise the formalism to e.g. planning with partial observability, non-determinism or epistemic planning.[5]

5 Belief states, partial observability, and conditional actions

Consider the birthday present example formalised as the propositional planning task of Example 4. Assume now that there is not only one, but two, local post offices, and the father does not know in which one the parcel is. We can assume *Loc* correspondingly contains two constants, PostOffice1 and PostOffice2. To represent this modified planning task we need two changes in the underlying formalism: *belief states* and *conditional actions*. We need belief states to represent the uncertainty of the agent. A **belief state** in this setting is a set of propositional states, that is, a set of subsets of P (where P is the set of atomic propositions). The initial belief state of our agent is now:

$$s_0 = \{\{\mathsf{At}(\mathsf{Father},\mathsf{Home}), \mathsf{At}(\mathsf{Present},\mathsf{PostOffice1})\}, \{\mathsf{At}(\mathsf{Father},\mathsf{Home}), \mathsf{At}(\mathsf{Present},\mathsf{PostOffice2})\}\}. \quad (2)$$

The state s_0 contains two propositional states, where the first one represents the situation where the present is in PostOffice1, and the second presents the situation where it is in PostOffice2. We define a propositional formula φ to be **true** in a belief state s, written $s \models \varphi$, if φ is true in all propositional states of s. Hence we have

(1) $s_0 \models \mathsf{At}(\mathsf{Father},\mathsf{Home})$

(2) $s_0 \not\models \mathsf{At}(\mathsf{Present},\mathsf{PostOffice1})$

[5] In certain cases the difference between expressing a planning task in the *lifted* first-order STRIPS representation and its induced propositionalisation becomes essential: for some of the complexity results measured in the size of the planning tasks; for practical convenience of representation; or e.g. for learning actions/action models [30, 24]. However, for the purposes of this paper, the grounded/propositionalised representation is sufficient.

(3) $s_0 \not\models \mathsf{At}(\mathsf{Present}, \mathsf{PostOffice2})$

(4) $s_0 \models \mathsf{At}(\mathsf{Present}, \mathsf{PostOffice1}) \vee \mathsf{At}(\mathsf{Present}, \mathsf{PostOffice2})$.

This represents the **internal perspective** of the father in the belief state s_0: He can verify (knows) that he is home (1) and can verify (knows) that the present is in PostOffice1 or PostOffice2 (4), but doesn't know which (2–3). Planning in the space of belief states rather than propositional states is called **planning under partial observability** (and planning on propositional states is then called **planning under full observability**). In the following, and in line with modal logic, we will call the elements of belief states **worlds**.

To represent the modified example we also need to allow *conditional actions*. The agent can attempt to pick up the present at either of the two post offices, but whether he is succesful is conditional on whether it is the correct one. Symmetric to the generalisation from representing states as subsets of P to sets of such subsets, we can generalise actions from being pairs $\langle pre(a), post(a)\rangle$ to be sets of such pairs. We can then represent the attempted pickup action by:

$$\begin{array}{l}\mathsf{TryPickUp}(agt, obj, from) = \{\\ \quad \langle \mathsf{At}(agt, from) \wedge \mathsf{At}(obj, from) \wedge \neg \mathsf{Has}(agt, obj), \mathsf{Has}(agt, obj) \wedge \neg \mathsf{At}(obj, from)\rangle,\\ \quad \langle \mathsf{At}(agt, from) \wedge \neg \mathsf{At}(obj, from), \top\rangle\\ \}\end{array} \tag{3}$$

where the postcondition $\top$ means that nothing changes. From now on we will, in line with the literature on dynamic epistemic logic, call pairs $\langle pre(e), post(a)\rangle$ **events**. So a conditional action like the one above is a set of events: a set of the possible things that might happen when the action is executed. The TryPickUp action above expresses that if the agent and the object are in the same location, the object will be successfully picked up (the first event of the action), and otherwise nothing will happen (the second event of the action).

Given a belief state s represented as a set of worlds and an action action a represented as a set of events, we can define a generalised transition function by

$$\gamma(s,a) = \{\gamma(w,e) \mid w \in s, e \in a, w \models pre(e)\}.^6 \tag{4}$$

Thus, e.g., where we abbreviate PostOffice1 by PO1, PostOffice2 by PO2 and Home by H:

$$\begin{array}{ll} s_1 & = \gamma(s_0, \mathsf{Go}(\mathsf{Father}, \mathsf{H}, \mathsf{PO1}))\\ & = \gamma(\{\{\mathsf{At}(\mathsf{Father}, \mathsf{H}), \mathsf{At}(\mathsf{Present}, \mathsf{PO1})\}, \{\mathsf{At}(\mathsf{Father}, \mathsf{H}), \mathsf{At}(\mathsf{Present}, \mathsf{PO2})\}\}, \mathsf{Go}(\mathsf{Father}, \mathsf{H}, \mathsf{PO1}))\\ & = \{\{\mathsf{At}(\mathsf{Father}, \mathsf{PO1}), \mathsf{At}(\mathsf{Present}, \mathsf{PO1})\}, \{\mathsf{At}(\mathsf{Father}, \mathsf{PO1}), \mathsf{At}(\mathsf{Present}, \mathsf{PO2})\}\}\\ \\ s_2 & = \gamma(s_1, \mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO1}))\\ & = \{\{\mathsf{At}(\mathsf{Father}, \mathsf{PO1}), \mathsf{Has}(\mathsf{Father}, \mathsf{Present})\}, \{\mathsf{At}(\mathsf{Father}, \mathsf{PO1}), \mathsf{At}(\mathsf{Present}, \mathsf{PO2})\}\}\\ \\ s_3 & = \gamma(s_2, \mathsf{Go}(\mathsf{Father}, \mathsf{PO1}, \mathsf{PO2}))\\ & = \{\{\mathsf{At}(\mathsf{Father}, \mathsf{PO2}), \mathsf{Has}(\mathsf{Father}, \mathsf{Present})\}, \{\mathsf{At}(\mathsf{Father}, \mathsf{PO2}), \mathsf{At}(\mathsf{Present}, \mathsf{PO2})\}\}\\ \\ s_4 & = \gamma(s_3, \mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO2}))\\ & = \{\{\mathsf{At}(\mathsf{Father}, \mathsf{PO2}), \mathsf{Has}(\mathsf{Father}, \mathsf{Present})\}\}\end{array}$$

[6] This is consistent with how the transition function is defined for conditional actions in [17], but only for actions in which the events have pairwise mutually inconsistent preconditions. If two events have mutually *consistent* preconditions, it means there exists states in which both are applicable. This can be interpreted in two ways. Either it represents *non-determinism* where only one of the events can actually take place when the action is executed. Or it represents a situation where multiple events occur in parallel. In [17], the latter interpretation is used. In this paper and in dynamic epistemic logic, the first interpretation is used.

$$\begin{aligned} s_5 &= \gamma(s_4, \mathsf{Go}(\mathsf{Father}, \mathsf{PO2}, \mathsf{Home})) \\ &= \{\{\mathsf{At}(\mathsf{Father}, \mathsf{Home}), \mathsf{Has}(\mathsf{Father}, \mathsf{Present})\}\} \end{aligned}$$

$$\begin{aligned} s_6 &= \gamma(s_5, \mathsf{Wrap}(\mathsf{Father}, \mathsf{Present})) \\ &= \{\{\mathsf{At}(\mathsf{Father}, \mathsf{Home}), \mathsf{Has}(\mathsf{Father}, \mathsf{Present}), \mathsf{Wrapped}(\mathsf{Present})\}\} \end{aligned}$$

Note how the belief state goes down from cardinality 2 to cardinality 1 when going from s_3 to s_4. At plan time, when deliberating about the possible action sequences, the father doesn't know whether he will have the present after having visited only PostOffice1, and he hence has to represent both possibilities. In s_4, however, he knows, even at plan time, that he will have the present, since if he didn't get it at PostOffice1, he will be sure to get it at PostOffice2. The calculations above show that a solution to the modified planning task is

$$\begin{aligned} \pi = \quad & \mathsf{Go}(\mathsf{Father}, \mathsf{H}, \mathsf{PO1}); \mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO1}); \mathsf{Go}(\mathsf{Father}, \mathsf{PO1}, \mathsf{PO2}); \\ & \mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO2}); \mathsf{Go}(\mathsf{Father}, \mathsf{PO2}, \mathsf{Home}); \mathsf{Wrap}(\mathsf{Father}, \mathsf{Present}). \end{aligned} \tag{5}$$

There are, unfortunately, several weaknesses in the current approach. An important weakness is that the formalism does so far not distinguish between between the kind of uncertainty the father has in s_1 and the kind of uncertainty he has in s_2. In s_1, the father has **run time uncertainty** [26] about the location of the present. This means that even at execution time, when the action $\mathsf{Go}(\mathsf{Father}, \mathsf{H}, \mathsf{PO1})$ has been executed in the initial state, the father still does not know which of the two worlds of s_1 is the actual one. In s_2, however, he should only have **plan time uncertainty** [26]. This means that when he is deliberating about the possible action sequences and their potential outcomes, he can not tell which of the two worlds of s_2 is going to be realised, but at execution time he *will* know. This is because attempting to pick up the present at PO1 has the side-effect of informing him whether the present is there or not. The distinction between plan time and run time uncertainty is not represented in our current formalism, since both kinds of uncertainty are modelled exactly the same: by simply having a set of worlds representing those situations that the agent cannot distinguish between (whether at plan time or at run time).

The simplest and most common solution to this problem in the automated planning literature (see e.g. [17, 28]) is to treat *observability* as a static partition on the set of possible worlds, so that certain possible worlds are always distinguishable and others never are. What becomes distinguishable at execution time is then hardcoded into this observability partition. This approach is however insufficient for our purposes. For instance, assume the present is initially located at PO2. Assume further that the father initially goes to PO1 to attempt picking it up and then goes home again. Then afterwards he will be in exactly the same world as initially, namely the world satisfying $\mathsf{At}(\mathsf{Father}, \mathsf{Home}) \wedge \mathsf{At}(\mathsf{Present}, \mathsf{PO2})$. But he will not be in the same *information state*: He learned that the present is not at PO1 (and therefore also learned that it must be at PO2). To be able to represent this in an appropriate way we need to take the final step into planning based on epistemic logic.[7]

[7] In principle, in the single-agent case, we could alternatively consider modelling states as sets of sets of worlds (that is, elements of 2^{2^P}) and actions a sets of sets of events. In a state $\{\{w_1^1, w_1^2, \ldots, w_1^{n_1}\}, \{w_2^1, w_2^2, \ldots, w_2^{n_2}\}, \ldots, \{w_m^1, w_m^2, \ldots, w_m^{n_m}\}\}$, two worlds w_i^k and w_j^l would then always be plan time indistinguishable, but only run time indistinguishable if $i = j$. This is essentially equivalent to representing states as models of single-agent epistemic logic (see next section). Since we are anyway going to generalise to multi-agent planning, we choose to move straight to epistemic logic, rather than consider this further intermediate step.

6 Epistemic logic

Let P be a finite set of atomic propositions (often we will take P to be the set of ground atoms of a function-free first-order language as in the previous sections). Let $\mathcal{A}$ be a finite set of agents. The **epistemic language** on $P, \mathcal{A}$, denoted $\mathcal{L}_{\mathrm{KC}}(P, \mathcal{A})$, is generated by the following BNF:

$$\varphi ::= \top \mid \bot \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi \mid C\varphi,$$

where $p \in P$ and $i \in \mathcal{A}$. We read $K_i\varphi$ as "agent i knows φ" and $C\varphi$ as "it is common knowledge that φ". When P and $\mathcal{A}$ are clear from the context (or no assumptions about them are made), we will write $\mathcal{L}_{\mathrm{KC}}$ for $\mathcal{L}_{\mathrm{KC}}(P, \mathcal{A})$. $\mathcal{L}_{\mathrm{K}}$ is the language $\mathcal{L}_{\mathrm{KC}}$ without the C modality.

Definition 4. Let P and $\mathcal{A}$ be as above. An **epistemic model** on $P, \mathcal{A}$ is $\mathcal{M} = (W, (\sim_i)_{i\in\mathcal{A}}, L)$ where

- The **domain** W is a non-empty finite set of **worlds**.
- $\sim_i \subseteq W \times W$ is an equivalence relation called the **indistinguishability relation** for agent $i \in \mathcal{A}$.[8]
- $L : W \to 2^P$ is a **labelling function** assigning a propositional valuation (a set of propositions) to each world.

For $W_d \subseteq W$, the pair $(\mathcal{M}, W_d)$ is called an **epistemic state** (or simply a **state**), and the worlds of W_d are called the **designated worlds**. A state is called **global** if $W_d = \{w\}$ for some world w (called the **actual world**), and we then often write $(\mathcal{M}, w)$ instead of $(\mathcal{M}, \{w\})$. We use $S^{\mathrm{gl}}(P, \mathcal{A})$ to denote the set of global states (or simply S^{gl} if P and $\mathcal{A}$ are clear from the context). For any state $s = (\mathcal{M}, W_d)$ we let $\mathit{Globals}(s) = \{(\mathcal{M}, w) \mid w \in W_d\}$. A state $(\mathcal{M}, W_d)$ is called a **local state** for agent i if W_d is closed under $\sim_i$ (that is, if $w \in W_d$ and $w \sim_i v$ implies $v \in W_d$). Given a state $s = (\mathcal{M}, W_d)$, the **associated local state** of agent i, denoted s^i, is $(\mathcal{M}, \{v \mid v \sim_i w \text{ and } w \in W_d\})$.

Definition 5. Let $(\mathcal{M}, W_d)$ be a state on $P, \mathcal{A}$ with $\mathcal{M} = (W, (\sim_i)_{i\in\mathcal{A}}, L)$. For $i \in \mathcal{A}$, $p \in P$ and $\varphi, \psi \in \mathcal{L}_{\mathrm{KC}}(P, \mathcal{A})$, we define truth as follows:

$$\begin{array}{lll}
(\mathcal{M}, W_d) \models \varphi & \text{iff} & (\mathcal{M}, w) \models \varphi \text{ for all } w \in W_d \\
(\mathcal{M}, w) \models p & \text{iff} & p \in L(w) \\
(\mathcal{M}, w) \models \neg\varphi & \text{iff} & \mathcal{M}, w \not\models \varphi \\
(\mathcal{M}, w) \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
(\mathcal{M}, w) \models K_i\varphi & \text{iff} & (\mathcal{M}, v) \models \varphi \text{ for all } v \sim_i w \\
(\mathcal{M}, w) \models C\varphi & \text{iff} & (\mathcal{M}, v) \models \varphi \text{ for all } v \sim^* w
\end{array}$$

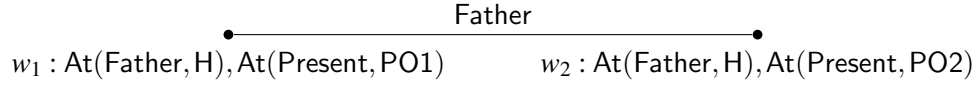
where $\sim^*$ is the transitive closure of $\bigcup_{i\in\mathcal{A}} \sim_i$.

Note that a formula φ is defined to be true in a non-global state $(\mathcal{M}, W_d)$ iff it is true in each of the global states $(\mathcal{M}, w)$, $w \in W_d$, it contains. This is consistent with our previous definition of truth in belief states as truth in all worlds of that state.

Example 5. Let $P = P_{\mathcal{L}}$, where $\mathcal{L}$ is the function-free first-order language of Example 3 extended with constant symbols $\mathsf{PostOffice1}$ and $\mathsf{PostOffice2}$ and let $\mathcal{A} = \{\mathsf{Father}\}$. Consider the initial state s_0 defined by (2). We can represent this via an epistemic model $\mathcal{M} = (W, \sim_{\mathsf{Father}}, L)$ on $P, \mathcal{A}$ with $W = \{w_1, w_2\}$, $\sim_{\mathsf{Father}} = W \times W$, $L(w_1) = \{\mathsf{At}(\mathsf{Father}, \mathsf{Home}), \mathsf{At}(\mathsf{Present}, \mathsf{PostOffice1})\}$ and $L(w_2) =$

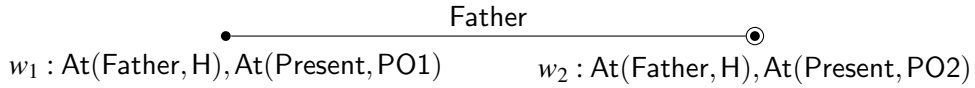
[8] We will later consider generalising to non-equivalence relations, but for now it is sufficient ot make this simplification and only consider equivalence relations.

$\{\mathsf{At}(\mathsf{Father},\mathsf{Home}),\mathsf{At}(\mathsf{Present},\mathsf{PostOffice2})\}$. Graphically, this epistemic model is represented as follows, using the abbreviations from earlier:



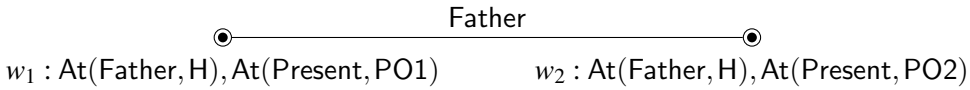
Here the nodes represent the worlds and the edges represent the indistinguishability relation (reflexive edges left out). Each node is labelled by its name and the set of atomic propositions true at the world. Consider the case where initially the present is at PostOffice2. This means that the actual world is w_2, and this situation can be represented by the global state $s_0 = (\mathcal{M}, w_2)$, that we graphically present as

$s_0 =$



We use ⊙ for designated worlds. The planning agent, Father, does not have access to this global state, since he considers it equally possible that the present is at PO1 (w_1 and w_2 are indistinguishable to him). The internal initial state of the father is his associated local state $s_0^{\mathsf{Father}} = (\mathcal{M}, W)$:

$s_0^{\mathsf{Father}} =$



We have $s \models \mathsf{At}(\mathsf{Present},\mathsf{PO2})$ but $s^{\mathsf{Father}} \not\models \mathsf{At}(\mathsf{Present},\mathsf{PO2})$: From the outside global perspective (the perspective of an omniscient external observer) it can initially be verified that the present is at PO2, but not from the perspective of the planning agent, Father, himself.

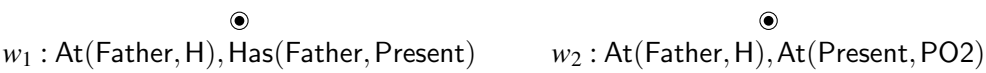
In general, any belief state $B = \{s_1,\ldots,s_n\}$ (set of propositional states) can be equivalently represented as an epistemic state $((W,\sim,L),W)$ with $W = \{w_1,\ldots,w_n\}$, $\sim = W \times W$, $L(w_i) = s_i$ for all $i = 1,\ldots,n$.

With epistemic models we can also capture the distinction between run time and plan time uncertainty. When the father contemplates on the possible action sequences to execute in his local state s^{Father}, the epistemic state representing the result of executing the action sequence

$$\mathsf{Go}(\mathsf{Father},\mathsf{H},\mathsf{PO1});\mathsf{TryPickUp}(\mathsf{Father},\mathsf{Present},\mathsf{PO1});\mathsf{Go}(\mathsf{Father},\mathsf{PO1},\mathsf{H})$$

will be

$s' =$



There is still two designated worlds: the agent do not at plan time know which one it will end up in. But they are now not indistinguishable (there is no indistinguishability edge connecting them): at run time, when the plan is executed, the agent will be able to tell whether it is in w_1 or w_2. In fact we have e.g. $s' \models K_{\mathsf{Father}}\mathsf{At}(\mathsf{Present},\mathsf{PO2}) \vee K_{\mathsf{Father}}\neg\mathsf{At}(\mathsf{Present},\mathsf{PO2})$: In s', the father knows whether the present is at PO2 or not. More generally, let $s = (\mathcal{M},W_d)$ be any local state of an agent $i \in \mathcal{A}$ and let $w_1,w_2 \in W_d$. The worlds w_1 and w_2 are called **run time indistinguishable** (for agent i) if $w_1 \sim_i w_2$. Otherwise, they are **plan time indistinguishable**.

The generalisation from belief states (in the standard AI sense of sets of propositional states [29]) to epistemic states (in the sense of epistemic logic) also allows us to represent planning tasks involving multiple agents. E.g. the father might want to make sure that his daughter doesn't get to know that he has a present for her (it is supposed to be a surprise). In this case, we could reformulate the goal as

$$\varphi_g = \mathsf{At}(\mathsf{Father},\mathsf{Home}) \wedge \mathsf{Has}(\mathsf{Father},\mathsf{Present}) \wedge \mathsf{Wrapped}(\mathsf{Present}) \wedge \neg K_{\mathsf{Daughter}}\mathsf{Has}(\mathsf{Father},\mathsf{Present}). \quad (6)$$

7 Dynamic epistemic logic

Generalising from propositional states to epistemic states amounted to generalise from propositional valuations to multisets of such valuations with an added indistinguishability relation for each agent. We can now apply the exact same trick to generalise from propositional actions to epistemic actions. A propositional action is an event $\langle pre(a), post(a)\rangle$, so an epistemic action will be a multiset of events with an indistinguishability relation for each agent. This is exactly how epistemic actions are defined in dynamic epistemic logic (DEL) with postconditions [12]. Such structures are called *event models* or *action models*.

Definition 6. An **action model** on $P,\mathcal{A}$ is $\mathcal{E} = (E, (\sim_i)_{i\in\mathcal{A}}, pre, post)$ where

- The **domain** E is a non-empty finite set of **events**.
- $\sim_i \subseteq E \times E$ is an equivalence relation called the **indistinguishability relation** for agent $i \in \mathcal{A}$.
- $pre : E \to \mathcal{L}_{\mathrm{KC}}(P,\mathcal{A})$ assigns a **precondition** to each event.
- $post : E \to \mathcal{L}_{\mathrm{KC}}(P,\mathcal{A})$ assigns a **postcondition** to each event. For all $e \in E$, $post(e)$ is a conjunction of literals over P (including the special atom $\top$).[9]

For $E_d \subseteq E$, the pair $(\mathcal{E}, E_d)$ is called an **epistemic action** (or simply an **action**), and the events in E_d are called the **designated events**. An action is called **global** if $E_d = \{e\}$ for some event e, and we then often write $(\mathcal{E}, e)$ instead of $(\mathcal{E}, \{e\})$. Similar to states, $(\mathcal{E}, E_d)$ is called a **local action** for agent i when E_d is closed under $\sim_i$. Given an action $a = (\mathcal{E}, E_d)$, the **associated local action** of agent $i \in \mathcal{A}$, denoted a^i, is $(\mathcal{E}, \{f \in E \mid f \sim_i e \text{ for some } e \in E_d\})$.

Any propositional action $a = \langle pre(a), post(a)\rangle$ trivially **induces** an epistemic action $(\mathcal{E}, \{e\})$ with $\mathcal{E} = (\{e\}, \{(e,e)\}, pre, post)$ where $pre(e) = pre(a)$ and $post(e) = post(a)$.

Example 6. We can now finally, using action models, give a satisfactory formal representation of the TryPickUp action of Section 5, cf. (3):

$\mathsf{TryPickUp}(agt, obj, from) =$

$$e_1 : \langle \mathsf{At}(agt,from) \wedge \mathsf{At}(obj,from) \wedge \neg\mathsf{Has}(agt,obj),\ \mathsf{Has}(agt,obj) \wedge \neg\mathsf{At}(obj,from)\rangle \qquad e_2 : \langle \mathsf{At}(agt,from) \wedge \neg\mathsf{At}(obj,from),\ \top\rangle$$

[9] This definition of postconditions is slightly less general than the standard definition of postconditions in DEL [12]. Normally, $post(e)$ is a mapping from atomic propositions to formulas $\mathcal{L}_{\mathrm{KC}}$, where $post(e)(p) = \varphi$ means that p after the event e has occurred gets whatever truth value φ had before e occurred. In [12] it is shown that any action model can be equivalently represented by one in which $post(e)(p) \in \{p, \top, \bot\}$ for all $e \in E$ and $p \in P$. This action model can potentially be exponentially larger. Such simplified (but potentially larger) action models can immediately be translated into the form defined here (see [6] for details). The advantages of the conventions of this paper is to have a more clear connection to the conventions of classical planning and to simplify notation.

We here use the same conventions for representing action models graphically as we did for epistemic models. Note that there is no indistinguishability edge for Father: He will at run time observe whether the action is succesful (e_1) or not (e_2). But at plan time he does not know which one it will be, which is why both events are designated.

The state-transition function of DEL is called *product update*, and is denoted by an infix $\otimes$ symbol. So instead of writing $\gamma(s,a)$, one writes $s \otimes a$.

Definition 7. Let a state $s = (\mathcal{M}, W_d)$ and an action $a = (\mathcal{E}, E_d)$ be given with $\mathcal{M} = (W, (\sim_i)_{i\in\mathcal{A}}, L)$ and $\mathcal{E} = (E, (\sim_i)_{i\in\mathcal{A}}, pre, post)$. Then the **product update** of s with a is $s \otimes a = ((W', (\sim'_i)_{i\in\mathcal{A}}, L'), W'_d)$ where

- $W' = \{(w,e) \in W \times E \mid (\mathcal{M}, w) \models pre(e)\}$
- $\sim'_i = \{((w,e),(w',e')) \in W' \times W' \mid w \sim_i w' \text{ and } e \sim_i e'\}$
- $L'(p) = (L(p) - \{p \mid \neg p \text{ is a negative literal of } post(e)) \cup \{p \mid p \text{ is a positive literal of } post(e)\}$
- $W'_d = \{(w,e) \in W' \mid w \in W_d \text{ and } e \in E_d\}$.

We say that an action $(\mathcal{E}, E_d)$ is **applicable** in a state $(\mathcal{M}, W_d)$ if for all $w \in W_d$ there is an event $e \in E_d$ such that $(\mathcal{M}, w) \models pre(e)$.

Note that this is the obvious generalisation to the epistemic setting of the state-transition function for belief states and sets of events given in (4). In particular, if s' is the epistemic state induced by a belief state s, and a' is the action model induced by a propositional action a, then $s' \otimes a'$ is the epistemic state induced by $\gamma(s,a)$.

Let $i \in \mathcal{A}$ be an agent, let $(\mathcal{M}, W_d)$ be a local state for i and let $(\mathcal{E}, E_d)$ be a local action for i. Then $(\mathcal{M}, W_d)$ and $(\mathcal{E}, E_d)$ represent agent i's view on a particular state and action. According to the definition above, the action $(\mathcal{E}, E_d)$ is then *applicable* in the state $(\mathcal{M}, W_d)$ if, for any of the worlds agent i considers possible (any world w in W_d), the action specifies at least one applicable event that agent i considers possible (an event e in E_d with $(\mathcal{M}, w) \models pre(e)$). In other words, the action is applicable from the perspective of agent i.

8 Epistemic planning tasks

We now have all the necessary ingredients for defining planning tasks in epistemic planning based on DEL. We simply define these as for propositional planning tasks, except we replace propositional actions by epistemic actions, propositional states by epistemic states and propositional goal formulas by epistemic goal formulas.

Definition 8. An **epistemic planning task** on $P, \mathcal{A}$ is (A, s_0, φ_g) where A is a finite set of epistemic actions on $P, \mathcal{A}$, s_0 is an epistemic state on $P, \mathcal{A}$, and $\varphi_g \in \mathcal{L}_{\mathrm{KC}}(P, \mathcal{A})$. We say that (A, s_0, φ_g) is an epistemic planning task **for agent** $i \in \mathcal{A}$ if s_0 and all $a \in A$ are local for i. A planning task (A, s_0, φ_g) is called **global** if s_0 is global. Given any planning task $\Pi = (A, s_0, \varphi_g)$, the **associated local planning task** of agent i, denoted Π^i, is $(\{a^i \mid a \in A\}, s_0^i, \varphi_g)$.[10]

An epistemic planning task (A, s_0, φ_g) **induces** a classical planning task $((S, A, \gamma), s_0, S_g)$ in a similar manner to propositional planning tasks:

[10] Given a planning task Π, the planning task Π^i is agent i's local perspective on that planning task. E.g. if the present is at PO1, then from an outside perspective, the planning task has an initial state where only the world satisfying At(Present, PO1) is designated, but agent Father is still forced to do planning based on the planning task Π^{Father} where both the At(Present, PO1)-world and the At(Present, PO2)-world are designated, since the father doesn't know which of the two is the actual initial state.

- $S = \{s_0 \otimes a_1 \otimes \cdots \otimes a_n \mid n \in \mathbb{N} \text{ and } a_1, \ldots, a_n \in A\}$
- $\gamma(s,a) = \begin{cases} s \otimes a & \text{if } a \text{ is applicable in } s \\ \text{undefined} & \text{otherwise} \end{cases}$
- $S_g = \{s \in S \mid s \models \varphi_g\}$.

As for propositional planning tasks, we can then define a **solution** to an epistemic planning task (A, s_0, φ_g) to be a solution to the induced classical planning task. This is equivalent to the existence of a sequence of actions $a_1; \cdots; a_n$ from A such that each a_i is applicable in $s_0 \otimes a_1 \otimes \cdots \otimes a_{i-1}$ $(i \leq n)$ and $s_0 \otimes a_1 \otimes \cdots \otimes a_n \models \varphi_g$.

Example 7. Let us provide a full formalisation of the birthday present example as an epistemic planning task. The epistemic action $\mathsf{TryPickUp}$ was presented in Example 6. The actions $\mathsf{Go}(agt, from, to)$ and $\mathsf{Wrap}(agt, obj)$ are simply the ones induced by their propositional counterparts. The initial state is the state s_0^{Father} of Example 5. The goal formula is $\varphi_g = \mathsf{At}(\mathsf{Father}, \mathsf{Home}) \wedge \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) \wedge \mathsf{Wrapped}(\mathsf{Present})$. Clearly the resulting planning task is a planning task for agent Father. We now get

$$s_1 = s_0^{\mathsf{Father}} \otimes \mathsf{Go}(\mathsf{Father}, \mathsf{H}, \mathsf{PO1}) = \begin{array}{ccc} \odot & \overset{\mathsf{Father}}{\rule{4em}{0.4pt}} & \odot \\ \mathsf{At}(\mathsf{Father}, \mathsf{PO1}), & & \mathsf{At}(\mathsf{Father}, \mathsf{PO1}), \\ \mathsf{At}(\mathsf{Present}, \mathsf{PO1}) & & \mathsf{At}(\mathsf{Present}, \mathsf{PO2}) \end{array}$$

$$s_2 = s_1 \otimes \mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO1}) = \begin{array}{cc} \odot & \odot \\ \mathsf{At}(\mathsf{Father}, \mathsf{PO1}), & \mathsf{At}(\mathsf{Father}, \mathsf{PO1}), \\ \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) & \mathsf{At}(\mathsf{Present}, \mathsf{PO2}) \end{array}$$

Note that the indistinguishability link between the two worlds is being cut when going from s_1 to s_2, since the two events of $\mathsf{TryPickUp}$ are distinguishable. We further get

$$s_3 = s_2 \otimes \mathsf{Go}(\mathsf{Father}, \mathsf{PO1}, \mathsf{PO2}) = \begin{array}{cc} \odot & \odot \\ \mathsf{At}(\mathsf{Father}, \mathsf{PO2}), & \mathsf{At}(\mathsf{Father}, \mathsf{PO2}), \\ \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) & \mathsf{At}(\mathsf{Present}, \mathsf{PO2}) \end{array}$$

$$s_4 = s_3 \otimes \mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO2}) = \begin{array}{cc} \odot & \odot \\ \mathsf{At}(\mathsf{Father}, \mathsf{PO2}), & \mathsf{At}(\mathsf{Father}, \mathsf{PO2}), \\ \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) & \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) \end{array}$$

$$s_6 = s_4 \otimes \mathsf{Go}(\mathsf{Father}, \mathsf{PO2}, \mathsf{H}) \otimes \mathsf{Wrap}(\mathsf{Father}, \mathsf{Present}) = \begin{array}{cc} \odot & \odot \\ \mathsf{At}(\mathsf{Father}, \mathsf{Home}), & \mathsf{At}(\mathsf{Father}, \mathsf{Home}), \\ \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) & \mathsf{Has}(\mathsf{Father}, \mathsf{Present}) \\ \mathsf{Wrapped}(\mathsf{Present}) & \mathsf{Wrapped}(\mathsf{Present}) \end{array}$$

Since the two worlds of s_6 have identical labels, we can take the *bisimulation contraction* which will preserve only one of them.[11] We clearly have $s_5 \models \varphi_g$. Hence the action sequence (5) above is a solution to the epistemic planning task. We could actually replace the $\mathsf{TryPickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO2})$ action in this plan with the epistemic action induced by the original non-conditional $\mathsf{PickUp}(\mathsf{Father}, \mathsf{Present}, \mathsf{PO2})$ action. This is because the father will know that the present is at PO2 when he gets there (since it wasn't at PO1).

[11] For a definition of bisimulation between epistemic models with multiple designated points, see [6]. For a definition of bisimulation on models of modal logic in general, and a definition of bisimulation contraction, see [4]. Bisimulation contractions are an indispensible tool in practice when working with product updates in DEL, since sequences of updates otherwise tend to produce very large model.

9 Conditional epistemic planning

The plan found in the previous example, Example 7, is clearly not always optimal. If the father gets the present already at PO1, there is no need to go to PO2 as well. So in this case the father will be able to reach the goal in 4 instead of 6 steps. But the father will of course not know this until run time (after having attempted to pick up the present at PO1). A sequential plan (sequence of actions) can not capture this, since it has a fixed number of actions, independent of the action outcomes. So we need to move to *conditional plans* (not to be confused with *conditional actions* that we already have). A conditional plan for the current planning task could e.g. be formulated as the following program

$$\begin{array}{l}\mathsf{Go}(\mathsf{Father},\mathsf{H},\mathsf{PO1});\mathsf{TryPickUp}(\mathsf{Father},\mathsf{Present},\mathsf{PO1});\\ \textbf{if } K_{\mathsf{Father}}\mathsf{Has}(\mathsf{Father},\mathsf{Present}) \textbf{ then } \mathsf{Go}(\mathsf{Father},\mathsf{PO1},\mathsf{H});\mathsf{Wrap}(\mathsf{Father},\mathsf{Present}) \textbf{ else}\\ \mathsf{Go}(\mathsf{Father},\mathsf{PO1},\mathsf{PO2});\mathsf{TryPickUp}(\mathsf{Father},\mathsf{Present},\mathsf{PO2});\mathsf{Go}(\mathsf{Father},\mathsf{PO2},\mathsf{H});\\ \mathsf{Wrap}(\mathsf{Father},\mathsf{Present})\end{array}$$

Such programs for epistemic planning tasks are formally defined and explored in [1] (single-agent case only).[12] Alternatively, such programs can be seen as abbreviations of PDL programs, e.g. the program **if** φ then π_1 **else** π_2 can be seen as shorthand for the PDL program $(\varphi?;\pi_1)\cup(\neg\varphi?;\pi_2)$. In [13], dynamic epistemic logic with postconditions and PDL constructs is studied (however not in a planning context). Programs like the one above are in [15] called *knowledge-based programs* and are there studied in an alternative logical setting. Knowledge-based programs are studied in a planning context in [23].

An alternative to conditional plans as programs is *policies*. A policy is a mapping from states into actions, that is, for each state the policy specifies the action chosen in that state (a policy is also sometimes called a *strategy* or a *protocol*, depending on the area). Since implemented planning systems often generate conditional plans via an AND-OR graph search in the state space, a policy comes as a more direct output of these algorithms than a program (e.g. one does not have to synthesise appropriate if-conditions). From now on we will only consider conditional plans as policies, not programs.

Definition 9. Let $\Pi = (A, s_0, \varphi_g)$ be a planning task and $i \in \mathcal{A}$ be an agent. An i-policy π for Π is a partial mapping $\pi : S^{\mathrm{gl}} \hookrightarrow A$ from global states into actions such that

(1) If $\pi(s) = a$ then a is applicable in s^i.

(2) If $s^i = t^i$ then $\pi(s) = \pi(t)$.

An i-policy is a policy from the perspective of agent i. Condition (1) ensures that in such a policy, agent i always knows that the next action to be performed is applicable (*knowledge of preconditions*). Condition (2) is a *uniformity condition*: If two global states are indistinguishable to agent i, agent i has to make the same choice in both.

Definition 10. An **execution** of a policy π from a global state s_0 is a maximal (finite or infinite) sequence of alternating global states and actions $(s_0, a_1, s_1, a_2, s_2, \dots)$ such that for all $m \geq 0$,

(1) $\pi(s_m) = a_{m+1}$, and

(2) $s_{m+1} \in \mathit{Globals}(s_m \otimes a_{m+1})$.

[12] Note that the if-condition of the if-then-else construct is a K-formula expressing knowledge of the planning agent (Father). All such if-conditions are required to be K-formulas of the planning agent, as an agent can only make a choice based on what he knows. Otherwise we could simply construct a conditional plan like **if** $\mathsf{At}(\mathsf{Present},\mathsf{PO1})$ **then** $\mathsf{Go}(\mathsf{Father},\mathsf{Home},\mathsf{PO1})$ **else** $\mathsf{Go}(\mathsf{Father},\mathsf{Home},\mathsf{PO2})$. However, in this case the agent would not know how to settle the truth-value of the if-condition $\mathsf{At}(\mathsf{Present},\mathsf{PO1})$ in the initial state, and would hence not know whether to execute $\mathsf{Go}(\mathsf{Father},\mathsf{Home},\mathsf{PO1})$ or $\mathsf{Go}(\mathsf{Father},\mathsf{Home},\mathsf{PO2})$.

An execution is called **successful** for a global planning task $\Pi = (A, s_0, \varphi_g)$ if it is a finite execution $(s_0, a_1, s_1, \ldots, a_n, s_n)$ such that $s_n \models \varphi_g$.

Definition 11. A policy π for a planning task $\Pi = (A, s_0, \varphi_g)$ is called a **solution** to Π if $\mathit{Globals}(s_0) \subseteq \mathrm{Dom}(\pi)$ and for each $s \in \mathrm{Dom}(\pi)$, any execution of π from s is successful for Π.[13]

Example 8. In the birthday present task we can now specify distinct actions to be performed in the two distinct global states of the state s_2 of Example 7 (the state after the father has attempted to pickup the present at PO1):

$$\pi(\ \overset{\circledcirc}{\mathsf{At}(\mathsf{Father},\mathsf{PO1}),\ \mathsf{Has}(\mathsf{Father},\mathsf{Present})}\quad \overset{\bullet}{\mathsf{At}(\mathsf{Father},\mathsf{PO1}),\ \mathsf{At}(\mathsf{Present},\mathsf{PO2})}\ ) = \mathsf{Go}(\mathsf{Father},\mathsf{PO1},\mathsf{Home})$$

$$\pi(\ \overset{\bullet}{\mathsf{At}(\mathsf{Father},\mathsf{PO1}),\ \mathsf{Has}(\mathsf{Father},\mathsf{Present})}\quad \overset{\circledcirc}{\mathsf{At}(\mathsf{Father},\mathsf{PO1}),\ \mathsf{At}(\mathsf{Present},\mathsf{PO2})}\ ) = \mathsf{Go}(\mathsf{Father},\mathsf{PO1},\mathsf{PO2}).$$

This partial policy can easily be extended into a full solution policy to the planning task.

Example 9. Consider extending the birthday present example with a post office employee, Employee, working at PO1. We can think of this as a new agent, so now $\mathcal{A} = \{\mathsf{Father}, \mathsf{Employee}\}$. It might be possible to call the employee from home to ask whether the present is at the post office. We can assume the employee knows whether the present is at his or her post office, so the initial state would be the same as in Example 7 (s_0^{Father}), except there is now reflexive loops for Employee at both worlds. We can now represent a general action of an agent i calling an agent j to ask whether a formula φ is true as the following action, where $i, j \in \mathcal{A}$ and $\varphi \in \mathcal{L}_{\mathrm{KC}}$:

$$\mathsf{Ask}(i,j,\varphi) = \quad \overset{\circledcirc}{\langle K_j\varphi, \top\rangle} \quad \overset{\circledcirc}{\langle K_j\neg\varphi, \top\rangle} \quad \overset{\circledcirc}{\langle \neg K_j\varphi \wedge \neg K_j\neg\varphi, \top\rangle}$$

The action model above corresponds to agent j giving a sincere answer to the question "is φ true?". Agent j saying "yes" to the question is represented by the event e_1, that is, it corresponds to an announcement that agent j knows φ. Event e_2 corresponds to saying "no", and e_3 corresponds to saying "I don't know". We now get

$$s_0^{\mathsf{Father}} \otimes \mathsf{Ask}(\mathsf{Father},\mathsf{Employee},\mathsf{At}(\mathsf{Present},\mathsf{PO1})) = \quad \overset{\circledcirc}{\mathsf{At}(\mathsf{Father},\mathsf{H}),\ \mathsf{At}(\mathsf{Present},\mathsf{PO1})} \quad \overset{\circledcirc}{\mathsf{At}(\mathsf{Father},\mathsf{Home}),\ \mathsf{At}(\mathsf{Present},\mathsf{PO2})}$$

This is because in w_1 of s_0^{Father} we have that $K_{\mathsf{Employee}}\mathsf{At}(\mathsf{Present},\mathsf{PO1})$ holds and in w_2 of s_0^{Father} we have $K_{\mathsf{Employee}}\neg\mathsf{At}(\mathsf{Present},\mathsf{PO1})$, and the action model for $\mathsf{Ask}(\mathsf{Father},\mathsf{Employee},\mathsf{At}(\mathsf{Present},\mathsf{PO1}))$

[13] Note that a solution policy has to lead to successful executions for all global states in s_0 (and for all global states along all possible executions of the policy). Such policies are often called *strong policies*, because the requirement is that they are guaranteed to succeed under alle circumstances (considered possible by the planning agent). In conditional planning, one often also considers weak solutions (*weak policies*) that have a possibility of succeeding, but might not always succeed. A weak solution to the birthday present task would be to go to PO1, attempt picking up the present there, go home, and then wrap the present if one has it. This plan can clearly fail if the present is at PO2. In this paper we restrict attention to strong solutions.

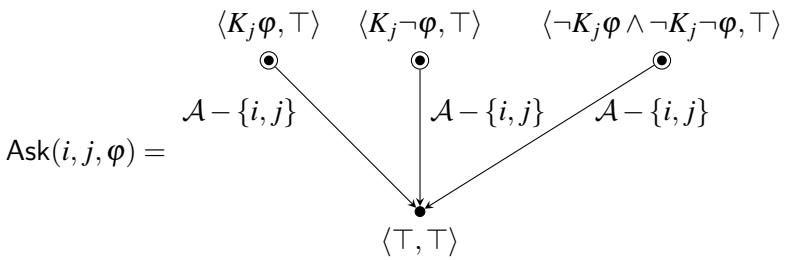
is clearly seen to cut the link between worlds satisfying $K_{\mathsf{Employee}}\neg\mathsf{At}(\mathsf{Present},\mathsf{PO1})$ and worlds satisfying $K_{\mathsf{Employee}}\neg\mathsf{At}(\mathsf{Present},\mathsf{PO1})$ (since there are two distinguishable events with these formulas as preconditions). Hence the planning agent, Father, can conclude

$$s_0 \otimes \mathsf{Ask}(\mathsf{Father},\mathsf{Employee},\mathsf{At}(\mathsf{Present},\mathsf{PO1})) \models K_{\mathsf{Father}}\mathsf{At}(\mathsf{Present},\mathsf{PO1}) \vee K_{\mathsf{Father}}\mathsf{At}(\mathsf{Present},\mathsf{PO2})$$

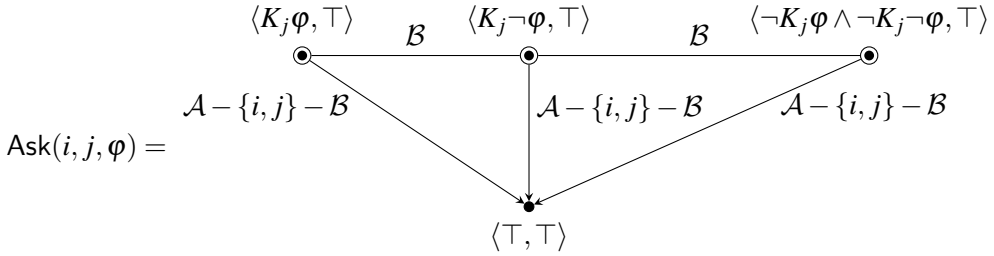
and hence that he can first call to ask the post office employee whether she has the present, and then next he will know where to go.

10 Public and private actions

The Ask action above is a bit simplified. It is **publicly observable**: the indistinguishability relation is the identity. When there is only two agents in the scenario this is acceptable, since they can only call each other, but if there were more agents present, these would probably not observe the phone call taking place. Consider adding a third agent, Employee2, working at PostOffice2. If Father calls Employee, Employee2 will not know, and might not even consider it possible that the phone call could have taken place. To model this we have to go beyond equivalence classes in epistemic models and action models, since now it becomes possible to have false beliefs, e.g. a false belief that no phone call took place (or a false belief that Father does not yet know $\mathsf{At}(\mathsf{Present},\mathsf{PO2})$). In this case, we could model the private phone call of agent i to agent j asking about φ as follows:



This models that agent i and j know the phone call takes place (and know the outcome of the phone call), but all other agents think that nothing has happened (the *skip* event $\langle\top,\top\rangle$). We can even model that some subset $\mathcal{B}\subseteq\mathcal{A}$ of agents hear the question being asked, but not the answer (they are together with i, but cannot hear what is being said in the other end):



These examples illustrate that designing action models to represent actions under multi-agent partial observability is far from a trivial matter. Some recent papers try to propose different ways of encoding observability information into a logical language, so that the action models themselves can become simpler, more uniform, and automatically induced from the underlying observability information [5, 7, 18]. However, there are still many unsolved problems in this area.

The possibility of representing *private actions* (only observed by a subset of agents, whereas the other agents believe nothing happens) is also what allows us to model the full version of the birthday present example, where the daughter does not get to know that the father has the present. We can for instance model that only agents who are *copresent* (in the same location) can observe actions taking place in that location (see [10] for a discussion and treatment of copresence in an epistemic planning setting). For instance, the Wrap action could be reformulated as follows:

$$\mathsf{Wrap}(agt, obj, loc) = \quad \overset{\{i : \neg \mathsf{At}(i, loc)\}_{i \in \mathcal{A}}}{\bullet \longrightarrow \bullet}$$
$$\langle \mathsf{At}(agt, loc) \wedge \mathsf{Has}(agt, obj) \wedge \neg \mathsf{Wrapped}(obj), \quad \langle \top, \top \rangle$$
$$\mathsf{Wrapped}(obj) \rangle$$

We here use a new type of action model that we call an **edge-conditioned action model** [5]. The meaning of the label on the edge is that there is an edge for agent i here if $\neg \mathsf{At}(i, loc)$ is true (agent i is not present in the location where the present is being wrapped). Such edge-conditioned action models are a variant of *generalised arrow updates* [22].

With the wrapping action presented as above, the father will e.g. be able to figure out that if the daughter is at home ($\mathsf{At}(\mathsf{Daughter}, \mathsf{Home})$ is true), then only the plan where he executes Wrap at the post office will lead to $\mathsf{Wrapped}(\mathsf{Present}) \wedge \neg K_{\mathsf{Daughter}} \mathsf{Has}(\mathsf{Father}, \mathsf{Present})$ (recall the state-transition system in Figure 2 that shows that the present can either be wrapped in the post office or at home).

11 Multi-agent planning

So far we have only considered the case of a single agent planning in the presence of other agents. The formal framework of course also allows to represent real multi-agent planning tasks. However, for true multi-agent planning a lot of additional design choices have to be made. Are the agents collaborating or competing? Are there *coalitions* of agents trying to achieve a joint goal, perhaps against the agents outside the coalition? Can the agents communicate and coordinate arbitrarily when coming up with a plan, and will they commit to such a joint plan? The papers [14, 8] look at *implicitly coordinated plans*, where agents have a joint goal but are not allowed to coordinate or negotiate in advance. All coordination should happen during plan executions as a result of announcements and observing the actions of others.

In the birthday present example it could e.g. be that Father calls Employee to announce the goal $\mathsf{Has}(\mathsf{Father}, \mathsf{Present})$. Assuming Employee is altruistic and adopts this as his or her own goal, Father and Employee are now engaged in planning with implicit coordination towards the joint goal $\mathsf{Has}(\mathsf{Father}, \mathsf{Present})$. Assume the present is at PostOffice2. Since Employee knows that Father doesn't know the location of the present, and that he cannot get the present unless he knows, she will choose to announce that the present is at PostOffice2. This is a case of implicit coordination: The father doesn't ask the employee about the location of the present, but the employee knows about the goal and can still plan to inform him, in order to allow him to plan the rest of the required actions. These ideas are explored further in [14, 8].

12 (Un)decidability and complexity

One of the most studied problems in epistemic planning based on DEL is the complexity of the plan existence problems. The **plan existence problem** for a class of planning tasks X is the following decision

problem: Given a planning task $\Pi \in X$, does there exist a solution to Π? So far only the complexity of deciding whether a sequential plan exists have been studied. For general epistemic planning tasks the problem is undecidable already with two agents, no common knowledge, and even without postconditions (that is, purely epistemic planning without ontic change) [2]. This has lead to a quest for finding decidable, but still reasonably expressive, fragments of epistemic planning. The first result along these lines proved that epistemic planning with propositional preconditions (that is, no epistemic formulas in the preconditions) is decidable [31]. An upper bound on complexity is in that paper shown to be $(n+1)$-EXPTIME for planning tasks in which the goal formula has modal depth n. In [11] it is shown that when the preconditions are propositional *and* there are no postconditions, then the plan existence problem is PSPACE-complete (for arbitrary goal formulas). If restricting further to certain types of actions, like private and public announcements, complexity of plan existence goes further down to NP-complete [9].

13 Alternative approaches to epistemic planning

Epistemic planning based on DEL takes a *semantic approach*, where states are represented as semantical objects, epistemic states. It is also possible to take a *syntactic approach*, where states are represented as knowledge-bases, sets of formulas known to be true. Interestingly, STRIPS and propositional planning are semantic and syntactic at the same time, since in propositional logic a semantic state is just a set of formulas (a set of true atomic propositions). This means that when generalising from STRIPS or propositional planning to epistemic planning, it is not immediately obvious whether a semantic or syntactic approach would be most appropriate. Both approaches have their strength and weaknesses.

Epistemic planning based on DEL also takes the approach of insisting on succinct action representations via action models. We argued in favour of this approach already very early in the paper by mentioning the weaknesses of attempting to represent planning tasks explicitly as state-transition systems. In fact, the state-transition systems induced by epistemic planning tasks will often be infinite, so even if choosing a state-transition system based approach, one will need a way to represent them finitely in order to be able to do planning based on them [20]. From the perspective of classical planning, the most appropriate approach to representing state-transition systems compactly seems to be representing the actions themselves in a compact way (using some kind of action models or action schemas).

Based on the above, we can distinguish approaches to epistemic planning along two dimensions:

- **Semantic approaches** versus **syntactic approaches**.
- **Action model based approaches** versus **state-transition system based approaches**.

The approach of this paper is, as mentioned, semantic and action model based. Syntactic approaches to epistemic planning can be found in e.g. the (single-agent) PKS planner [27], the (multi-agent) planning framework of [25] and the compilation approach of [21], translating a restricted fragment of epistemic planning into classical planning. The state-transition system based approach is found in a number of papers in temporal epistemic logic, e.g. [19, 20]. In these papers, the planning domains are represented by a type of epistemic state-transition system called a *concurrent epistemic game structure* (CEGS). Having an explicitly represented state-transition system like a CEGS tends to make it easier to define complex notions of multi-agent plans (strategies). On the other hand, by working directly with state-transition systems, some of the important problems of epistemic planning are being silently bypassed. These problems include compact representations of the state-transition system, and efficient heuristics for avoiding to build the entire state-transition system when planning. They also include problems of how to provide a general approach to multi-agent observability and problems of concurrent composition

of actions of multiple agents. In CEGS, transitions represent *joint actions*, one action per agent. The problem of how to make a parallel composition of the actions of the individual agents has to be solved before one can even build the CEGS for a given planning domain. In epistemic planning based on DEL, this problem has to be solved at the level of action models: How do we make a parallel composition of two action models in case they are not independent (e.g. two agents trying to open a door at the same time, but from opposite sides). Epistemic planning tasks can be seen as inducing e.g. forests of epistemic temporal logic [3] or CEGSs. The exact relations have not been explored yet, but providing a link between the action model based approaches and the state-transition system based approaches could be very valuable and provide a possibility of getting the best of both worlds.

Similarly, providing links and connections between syntactic and semantic approaches seem to be potentially very valuable. In a semantic approach, one is essentially modelling *ignorance*: the more ignorance, the bigger the state. In a syntactic approach, one is essentially modelling *knowledge*: the more knowledge, the bigger the state. Knowledge and ignorance are each others duals, and hence the semantic and the syntactic approach are also each others duals. It seems that when humans are planning, we are sometimes using a more semantic approach, and sometimes a more syntactic one, depending on the planning task at hand. It would hence also be very interesting to see whether planning frameworks could be developed that would employ or combine both approaches.

References

[1] Mikkel Birkegaard Andersen, Thomas Bolander & Martin Holm Jensen (2012): *Conditional Epistemic Planning*. *Lecture Notes in Artificial Intelligence* 7519, pp. 94–106, doi:10.1007/978-3-642-33353-8_8. Proceedings of JELIA 2012.

[2] Guillaume Aucher & Thomas Bolander (2013): *Undecidability in Epistemic Planning*. In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 27–33.

[3] Johan van Benthem, Jelle Gerbrandy & Eric Pacuit (2007): *Merging frameworks for interaction: DEL and ETL*. In: *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge*, TARK '07, ACM, New York, NY, USA, pp. 72–81, doi:10.1145/1324249.1324262.

[4] P. Blackburn, M. de Rijke & Y. Venema (2001): *Modal Logic*. *Cambridge Tracts in Theoretical Computer Science* 53, Cambridge University Press, Cambridge, UK, doi:10.1017/CBO9781107050884.

[5] Thomas Bolander (2014): *Seeing is Believing: Formalising False-Belief Tasks in Dynamic Epistemic Logic*. In Andreas Herzig & Emiliano Lorini, editors: *Proceedings of the European Conference on Social Intelligence (ECSI-2014)*, *CEUR Workshop Proceedings* 1283, CEUR-WS.org, pp. 87–107.

[6] Thomas Bolander & Mikkel Birkegaard Andersen (2011): *Epistemic Planning for Single- and Multi-Agent Systems*. *Journal of Applied Non-Classical Logics* 21, pp. 9–34, doi:10.3166/jancl.21.9-34.

[7] Thomas Bolander, Hans van Ditmarsch, Andreas Herzig, Emiliano Lorini, Pere Pardo & François Schwarzentruber (2015): *Announcements to Attentive Agents*. *Journal of Logic, Language and Information*, pp. 1–35, doi:10.1007/s10849-015-9234-3.

[8] Thomas Bolander, Thorsten Engesser, Robert Mattmller & Bernhard Nebel (2016): *Better Eager Than Lazy? How Agent Types Impact the Successfulness of Implicit Coordination*. In: *Distributed and Multi-Agent Planning (DMAP-16)*, pp. 42–49.

[9] Thomas Bolander, Martin Holm Jensen & François Schwarzentruber (2015): *Complexity Results in Epistemic Planning*. In Qiang Yang & Michael Wooldridge, editors: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, AAAI Press, pp. 2791–2797.

[10] Michael Brenner & Bernhard Nebel (2009): *Continual planning and acting in dynamic multiagent environments*. Autonomous Agents and Multi-Agent Systems 19(3), pp. 297–331, doi:10.1007/s10458-009-9081-1.

[11] Tristan Charrier, Bastien Maubert & François Schwarzentruber (2016): *On the Impact of Modal Depth in Epistemic Planning*. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, USA, July 12-15, 2016.

[12] Hans van Ditmarsch & Barteld Kooi (2008): *Semantic Results for Ontic and Epistemic Change*. In Giacomo Bonanno, Wiebe van der Hoek & Michael Wooldridge, editors: Logic and the Foundation of Game and Decision Theory (LOFT 7), Texts in Logic and Games 3, Amsterdam University Press, pp. 87–117.

[13] Jan van Eijck (2014): *Dynamic epistemic logics*. In: Johan van Benthem on Logic and Information Dynamics, Springer, pp. 175–202.

[14] Thorsten Engesser, Thomas Bolander, Robert Mattmller & Bernhard Nebel (2017): *Cooperative Epistemic Multi-Agent Planning for Implicit Coordination*. In: Proceedings of Methods for Modalities, Electronic Proceedings in Theoretical Computer Science.

[15] Ronald Fagin, Joseph Y. Halpern, Yoram Moses & Moshe Y. Vardi (1995): *Reasoning About Knowledge*. MIT Press.

[16] R. Fikes & N. Nilsson (1971): *STRIPS: A new approach to the application of theorem proving to problem solving*. Artificial Intelligence 2, pp. 189–203, doi:10.1016/0004-3702(71)90010-5.

[17] Malik Ghallab, Dana S. Nau & Paolo Traverso (2004): *Automated Planning: Theory and Practice*. Morgan Kaufmann.

[18] Andreas Herzig, Emiliano Lorini & Faustine Maffre (2015): *A poor mans epistemic logic based on propositional assignment and higher-order observation*. In: Logic, Rationality and Interaction, Lecture Notes in Computer Science 9394, Springer, doi:10.1007/978-3-662-48561-3_13.

[19] Wiebe van der Hoek & Michael Wooldridge (2002): *Tractable Multiagent Planning for Epistemic Goals*. In: In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), ACM Press, pp. 1167–1174, doi:10.1145/545056.545095.

[20] Wojciech Jamroga & Thomas Aagotnes (2007): *Constructive knowledge: what agents can achieve under imperfect information*. Journal of Applied Non-Classical Logics 17(4), pp. 423–475, doi:10.3166/jancl.17.423-475.

[21] Filippos Kominis & Hector Geffner (2014): *Beliefs in multiagent planning: From one agent to many*. In: Proc. ICAPS Workshop on Distributed and Multi-Agent Planning.

[22] Barteld Kooi & Bryan Renne (2011): *Generalized arrow update logic*. In: Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge, ACM, pp. 205–211, doi:10.1145/2000378.2000403.

[23] Jérôme Lang & Bruno Zanuttini (2013): *Knowledge-Based Programs as Plans: Succinctness and the Complexity of Plan Existence*. In: TARK 2013.

[24] Kira Mourão, Luke S. Zettlemoyer, Ronald P. A. Petrick & Mark Steedman (2012): *Learning STRIPS Operators from Noisy and Incomplete Observations*. In Nando de Freitas & Kevin P. Murphy, editors: Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012, AUAI Press, pp. 614–623.

[25] Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian R Pearce & Liz Sonenberg (2015): *Planning Over Multi-Agent Epistemic States: A Classical Planning Approach (Amended Version)*. In: Distributed and Multi-Agent Planning (DMAP-15), pp. 60–67, doi:10.1.1.684.2394.

[26] Ronald P. A. Petrick & Fahiem Bacchus (2002): *A Knowledge-Based Approach to Planning with Incomplete Information and Sensing*. In Malik Ghallab, Joachim Hertzberg & Paolo Traverso, editors: Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002), AAAI Press, Menlo Park, CA, pp. 212–221.

[27] Ronald P. A. Petrick & Fahiem Bacchus (2004): *PKS: Knowledge-Based Planning with Incomplete Information and Sensing*. In: *ICAPS 2004*.

[28] Jussi Rintanen (2006): *Introduction to automated planning*.

[29] Stuart Russell & Peter Norvig (1995): *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[30] Thomas J. Walsh & Michael L. Littman (2008): *Efficient Learning of Action Schemas and Web-service Descriptions*. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, AAAI Press, pp. 714–719.

[31] Quan Yu, Ximing Wen & Yongmei Liu (2013): *Multi-agent epistemic explanatory diagnosis via reasoning about actions*. In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 27–33.