An Epistemic Planning System Based on the Event Calculus

Manfred Eppe, Frank Dylla

SFB/TR 8 Report No. 033-11/2012
Report Series of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition
Universität Bremen / Universität Freiburg

Contact Address:

Dr. Thomas Barkowsky
SFB/TR 8
Universität Bremen
P.O.Box 330 440
28334 Bremen, Germany

Tel +49-421-218-64233
Fax +49-421-218-64239
barkowsky@sfbtr8.uni-bremen.de
www.sfbtr8.uni-bremen.de

© 2012 SFB/TR 8 Spatial Cognition

An Epistemic Planning System Based on the Event Calculus

Manfred Eppe[1] and Frank Dylla[2]
[1]*Department of Computer Science and Mathematics, University of Bremen, Germany*
meppe@informatik.uni-bremen.de

[2]Cognitive Systems, SFB/TR8 Spatial Cognition, University of Bremen, Germany
dylla@informatik.uni-bremen.de

Keywords: Planning : Epistemic Reasoning : Action Theory : Event Calculus

Abstract: We propose an approach for single-agent epistemic planning in domains with incomplete knowledge. We argue that on the one hand the integration of epistemic reasoning into planning is useful because it makes the use of sensors more flexible. On the other hand, defining an epistemic problem description is an error prone task as the epistemic effects of actions are more complex than their usual physical effects. To overcome this problem we apply the axioms of the Discrete Event Calculus Knowledge Theory (DECKT) as rules to compile non-epistemic planning problem descriptions into epistemic descriptions. We show how the resulting planning problems are solved and describe our implemented prototype which is based on Answer Set Programming (ASP).

1 INTRODUCTION

Many approaches to AI planning rely on the strong and unrealistic assumption that complete knowledge about the world is available. A good indicator for this is that the standardized Problem Domain Definition Language (PDDL) (Mcdermott et al., 1998) contains no standardized constructs to express incomplete knowledge or sensing.

There are planning systems for domains with incomplete knowledge, e.g. (Brenner and Nebel, 2009) or (Pryor and Collins, 1996), but these systems usually do not consider conditional effects of actions and the information acquired by sensing is always used directly.[1] We point out that if epistemic reasoning (ER) (Blackburn et al., 2001) is integrated in planning, then additional information, i.e. other than only the measured value, can be obtained by sensing. This leads us to our first hypothesis: *A planning system which accounts for both epistemic reasoning and actions with conditional effects can be used to compensate for missing or broken sensors and to work around expensive sensing actions using cheaper sensing actions.* For example, if one wants to find out whether a liquid is poisonous it can be costly to learn about its poisonousness by drinking the liquid.

The hypothesis is based on the idea that conditional effects of actions can be exploited to achieve *indirect sensing*. For example, assume an agent with the (sub-)goal to know whether a door is open or not. If equipped with an appropriate sensor it will sense the door state and the goal is achieved. But how can a robot achieve the goal if it doesn't feature a specific door sensor or if the sensor is broken? In fact, it can try to use other sensor information to acquire the door state. For example, if the robot is equipped with a location sensor it can infer the door state indirectly by means of this sensor: send the robot through the door,[2] sense its location and infer the open-state of the door via the robot's new location. If it is behind the door then the door is open and if it is still in front of the door, then the door is closed. This means, the robot can deduce the door state by executing an action and evaluating its effect(s).

Epistemic reasoning specifically deals with such kinds of problems. In the following, we use the term *epistemic planning* (EP) when we speak about a single agent which is aware of having incomplete knowledge about the world, which can sense to acquire knowledge, and which integrates three inference types in the planning process:

1. Infer knowledge loss due to uncertain action ef-

[1]A notable exception is (Petrick and Bacchus, 2004) which we discuss later.

[2]We assume that actions return an 'executed' flag without giving any information about the success.

fects. (After executing the move-action, if the robot does not know whether the door is open, then knowledge about its location is lost.)

2. Infer knowledge about conditions of actions through effects. (If after moving the robot is behind the door, then the door must have been open.)
3. Infer knowledge about effects of actions through conditions. (If after moving the robot comes to know that the door is open, then it can infer that it must be behind the door.)

For further aspects of epistemic reasoning we refer to Blackburn et al. (2001).

A problem in EP is that deriving epistemic representations of planning problems can be cumbersome when done manually. Knowledge-level effects of actions must be given explicitly in the form of implications as they can not be described as propositional literals anymore. For example, consider the action of moving through a door again. A knowledge-level conditional effect of this action is: If the door's open-state is unknown, then the following implication holds: *If the robot is behind door, then the door is/was open.* In Section 4.1 we point out that for each non-epistemic conditional action effect with $|C|$ conditions, $2 \cdot |C|$ epistemic effects have to be specified. Further, if one does not include certain conditional effects in the specification of an action, then the resulting planning problem representation is not sound wrt. epistemic theory. For example, consider the move-through-door-action again. It is not epistemically accurate if it does not consider loss of knowledge about the robot's position when the open-state of a door is unknown.

Based on these thoughts we formulate our second hypothesis: *The automated translation of non-epistemic planning problem domains into epistemic ones simplifies the formal description of planning problems.* We argue that automated translation also guarantees soundness wrt. epistemic theory. We further make the claim that currently there exists no planning system which automatically generates epistemic planning domain descriptions and which considers actions with conditional effects. Consequently, no existing planning system can guarantee the soundness property of an epistemic problem specification wrt. epistemic theory. Instead, the soundness-property of existing approaches relies on the manually defined epistemic action descriptions.

Non-epistemic problem descriptions are given in terms of "how the world is" and need to be redefined in terms of "what an agent knows about the world". This comprises the initial state, the goals, the preconditions of actions, and the conditional effects of actions such that they account for the three knowledge-level effects mentioned above.

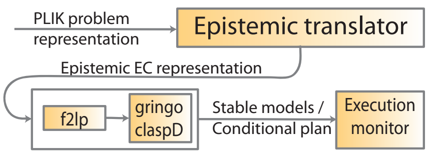


Figure 1: The toolchain of our planning system

We propose to automate this redefinition and implement a translation tool. The tool is part of the planning system depicted in Figure 1: A non-epistemic planning problem description is defined in our Planning Language for Incomplete Knowledge (PLIK). We develop this language to simplify the definition of planning problems (see Section 3.1 and 3.2). The PLIK-representation of a planning problem is translated into an epistemic dialect of the Event Calculus (EC) (Kowalski, 1986) using the Discrete Event Calculus Knowledge Theory (DECKT) (Patkos and Plexousakis, 2009). The epistemic problem description is input to the f2lp-tool by Lee and Palla (2009, 2012) which again translates the first order logical problem description into an answer set programming (ASP) problem. The solutions to the ASP problem are stable models (Gelfond and Lifschitz, 1988) which we interpret as conditional plans. Before detailing our approach we introduce some fundamentals and related work.

2 RELATED WORK

We introduce related work from the field of Automated Planning and Scheduling and the field of Action Theories and Epistemic Reasoning. In literature, the term "planning" is not always used coherently. In our context, we understand planning as the task of finding an admissible sequence of actions which leads from an initial state to a goal state (Russell and Norvig, 2002).

Automated Planning and Scheduling with Incomplete Knowledge The task of generating a plan which involves sensing actions and incomplete knowledge is well-known as Contingent Planning (CP). Examples are (Pryor and Collins, 1996), (Hoffmann and Brafman, 2005), (Eiter et al., 2000) and the Model Based Planning (MBP) system system by Bertoli et al. (2001). MBP is closely related to our work, as it implicitly accounts for the knowledge-level effects of actions. That means, one can specify conditional effects of actions in its input language NPDDL (Bertoli et al., 2002) and the planner handles epistemic effects. However, the authors do not provide a proof that the underlying action theory of their planner is epistemically sound.

The PKS planner by Petrick and Bacchus (2004) is currently the only system we know about which explicitly regards for knowledge-level effects of actions. Nevertheless, knowledge-level effects must be handled manually and demand a complex problem specification.

Another related research field which in particular covers non-deterministic effects of actions is probabilistic planning (Kushmerick et al., 1995). Our approach does not support to specify a numerical value as the probability for the outcome of an action or an initial world state. Instead, it focuses on the integration of epistemic reasoning into the planning process.

Action Theories and Epistemic Reasoning Action theories describe properties of the world and how they change. Prominent examples are the Situation Calculus (SC) (Reiter, 2001), Event Calculus (EC) (Kowalski, 1986), Temporal Action Logic (TAL) (Doherty et al., 1998) and Fluent Calculus (FC) (Thielscher, 1998). It is well known that these theories can be used for planning (Shanahan, 2000), (Kvarnström, 2005).

Action theories involving epistemic reasoning are introduced by Moore (1985), who describes the possible-worlds semantics of knowledge using Kripke structures and an epistemic *K*-fluent. Scherl and Levesque (2003) continue this work and solve the frame problem for epistemic SC. The PKS planner uses the epistemic SC as a formal basis.

IndiGolog is a high-level programming language by De Giacomo and Levesque (1998) which has a search-operator that can also be used to perform planning. Sardina et al. (2004) present a generalization of this search operator such that all plans are guaranteed to achieve the goal. Thus, the focus of their work is more on the avoidance of plans containing dead ends. However, in the same paper, the authors introduce the notion of an *epistemically accurate theory*: A theory is epistemically accurate if world states and action specifications are provided in what the agent knows about the world, in such a way that the knowledge-level effects of actions correctly reflect epistemic theory (Blackburn et al., 2001). While in IndiGolog the programmer is responsible for generating an epistemically accurate theory, theories derived from a PLIK representation are *guaranteed* to be epistemically accurate.

For FC a high-level programming language interpreter FLUX is available (Thielscher, 2005). Like IndiGolog, FLUX does not guarantee that manually implemented action effect axioms are epistemically accurate.

Another series of work on epistemic planning comes from the Dynamic Epistemic Logic (DEL) community (van Ditmarsch et al., 2007). For example, Bolander and Birkegaard Andersen (2011) show that DEL based EP is decidable for the single-agent case. However, DEL does not account for actions with conditional effects, so indirect knowledge-level effects of actions which we exploit in our work do not occur.

2.1 Theoretical Foundations

Our work is based the Event Calculus (EC) by Kowalski (1986) and the Discrete Event Calcululs Knowledge Theory (DECKT) by Patkos and Plexousakis (2009).

Event Calculus: EC is a formalization to reason about action and change. The Discrete Event Calculus (DEC) by Mueller (2005) is the particular dialect used in our work. The theory uses the predicate $\mathit{HoldsAt}\,(f, t)$to state that a fluent f holds at time t. $\mathit{ReleasedAt}\,(f, t)$ states that f is released from inertia at t. $\mathit{Happens}\,(e, t)$ denotes that the event e happens at t. $\mathit{Initiates}\,(e, f, t)$, $\mathit{Terminates}\,(e, f, t)$ and $\mathit{Releases}\,(e, f, t)$ define effects of events.[3] As usual in EC related literature, we use Δ to denote conjunctions of *Happens*-statements, i.e. plans, and γ to denote conjunctions of *HoldsAt*-statements. An *effect axiom* has the form $\gamma \Rightarrow \pi(e, f, t)$ where π represents either *Initiates*, *Terminates* or *Releases*. A *precondition axiom* has the form $\mathit{Happens}\,(e, t) \Rightarrow \gamma$, saying that an event can only happen if condition γ holds.[4]

Planning in EC is abductive reasoning. Shanahan (2000) describes non-epistemic EC planning as follows: Consider Γ to be the initial world state, Γ' the goal state and Σ a set of action specifications. One is interested in finding a plan Δ such that:

$$\begin{aligned} &\mathit{CIRC}[\Sigma; \mathit{Initiates, Terminates, Releases}] \wedge \\ &\qquad \mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Gamma \wedge \Omega \wedge \mathit{EC} \models \Gamma' \end{aligned} \quad (1)$$

[3]Throughout this text, all variables are implicitly universally quantified if not stated otherwise. Variables for events are denoted with e. We will use the term *event* and *action* interchangeably. Variables for fluents are denoted with f, for literals with l, for reified fluent formulae with ϕ and variables of the sort time with t. Note that reified fluent formulae ϕ do not contain quantifiers or predicates, only fluents. Thus, second order expressions do not occur.

[4]The implication is to be understood as "If e happens, then γ must be entailed in the theory". EC axioms also regulate which fluents can not hold, i.e. which γ can impossibly be entailed in the theory. Therefore an implication with *Happens* in the body can not trigger a term γ to be entailed in the theory and thus γ is a precondition as it restricts when an event e can happen. For details on precondition axioms in EC we refer to Mueller (2005).

where *CIRC* denotes circumscription of Σ and Δ and as described in (Mueller, 2005, frame problem) and Ω denotes uniqueness of names axioms.

Discrete Event Calculus Knowledge Theory: Patkos and Plexousakis (2009) developed DECKT, an epistemic theory for EC. They show that the theory is sound and complete wrt. **T** system (Blackburn et al., 2001) of the possible world semantics.They introduce an epistemic *Knows*-fluent using nested reification.[5] For example, $\mathit{HoldsAt}\,(\mathit{Knows}(\neg f), t)$ means that at time t the agent knows that f is false. *Knows*-fluents are released from inertia at all times in DECKT:

$$\mathit{ReleasedAt}\,(\mathit{Knows}(\phi), t) \tag{2}$$

DECKT also uses a fluent *KP(f)* which states that f is known persistently, i.e. *KP*-fluents are not released from inertia. DECKT states, that everything which is *KP*-known is also *Knows*-known:

$$\mathit{HoldsAt}\,(\mathit{KP}(\phi), t) \Rightarrow \mathit{HoldsAt}\,(\mathit{Knows}(\phi), t) \tag{3}$$

The nested reification in DECKT makes it possible to express strong negation (e.g. $\mathit{HoldsAt}\,(\mathit{Knows}(\neg f), t)$) and classical negation (e.g. $\neg\mathit{HoldsAt}\,(\mathit{Knows}(f), t)$). It also allows for expressing so-called Hidden Causal Dependencies (HCD), which are implications like $\mathit{HoldsAt}\,(\mathit{KP}(f \Rightarrow f'), t)$. This HCD expresses that it is known that if f is true then f' is also true. We use HCDs to formalize indirect sensing.

DECKT uses a knowledge-minimization axiom stating that a fluent can only be known if it is *KP*-known or if it is known through an implication. This is expressed with the axiom:

$$\begin{aligned}&\mathit{HoldsAt}\,(\mathit{Knows}(\phi), t) \Rightarrow \mathit{HoldsAt}\,(\mathit{KP}(\phi), t) \vee \\ &(\mathit{HoldsAt}\,(\mathit{KP}(\phi'), t) \wedge \mathit{HoldsAt}\,(\mathit{KP}(\phi' \Rightarrow \phi), t))\end{aligned} \tag{4}$$

In summary, DECKT can be seen as a set of axioms which formally relate the physical effects of actions to their knowledge-level effects. In our work, we use these axioms as rules for the translation of non-epistemic planning domain definitions to epistemic planning-domain definitions, thereby guaranteeing soundness of our work wrt. possible worlds semantics. In particular, this implies that the agent's knowledge about the world cannot be wrong:

$$\mathit{HoldsAt}\,(\mathit{Knows}(\phi), t) \Rightarrow \mathit{HoldsAt}\,(\phi, t) \tag{5}$$

In the following, we call the conjunction of DECKT and EC the Epistemic Event Calculus (EEC).

[5]In principle, introspection as in the S4 or S5 epistemic system would be possible with DECKT but is computationally expensive due to more deeply nested reification. However, for our purpose we do not need introspection.

3 EPISTEMIC PLANNING

We design a language called Planning Language for Incomplete Knowledge (PLIK) to represent planning problems. PLIK is a macro-language for EC and DECKT. We translate the PLIK representation into an epistemic EC-based representation and use a chain of ASP tools to generate a set of stable models which we interpret as a conditional plan.

3.1 Epistemic Planning Domains

We develop PLIK on top of EC to avoid EC's circumstantial syntax. As in usual non-epistemic EC planning, a problem specification consists of a set of types $\mathcal{T}$, a set of objects $\mathcal{O}$, a set of fluents $\mathcal{F}$, a set of action specifications $\mathcal{A}$ and a set of goals $\mathcal{G}$. Instead of a complete definition of the initial world state, PLIK's set of statements about the agent's initial knowledge $\mathcal{I}$ may be incomplete. $\mathcal{T}$, $\mathcal{O}$, $\mathcal{F}$, $\mathcal{A}$, $\mathcal{G}$ and $\mathcal{I}$ are finite and may be empty.

Types are *sorts* in an EC domain description. An example for a PLIK type specification is:

```
(:types Door Room Robot)
```

Objects are elements of a certain sort:

```
(:objects
  corridor,livingRoom - Room
  d1 - Door
  wc,vc - Robot )
```

where the "-" is to be read as "element-of".

Fluents have a set of arguments of a certain type. An example for the fluent specification in PLIK is:

```
(:fluents
  hasDoor(Room,Door)
  inRoom(Robot,Room) :functional
  opened(Door)
  isRobust(Robot) )
```

For functional fluents, defined by `:functional`, which have the arity a we add:

$$\begin{aligned}&\mathit{HoldsAt}\,(f(x_1,\ldots,x_{a-1},v), t) \wedge v \neq v' \\ &\qquad \Rightarrow \neg\mathit{HoldsAt}\,(f(x_1,\ldots,x_{a-1},v'), t)\end{aligned} \tag{6}$$

This assures that actions which affect functional fluents are specified correctly, e.g. the location of an object.

There may exist a sensing action which reveals the truth-value of a fluent. Therefore we define two deterministic events for each fluent f, *senseT(f)* and *senseF(f)*. These are used in Section 3.2 for specifying observations.

$$\begin{array}{l} \textit{Initiates}\,(senseT(f), KP(f), t) \\ \textit{Terminates}\,(senseT(f), KP(\neg f), t) \\ \textit{Initiates}\,(senseF(f), KP(\neg f), t) \\ \textit{Terminates}\,(senseF(f), KP(f), t) \end{array} \tag{7}$$

If f is functional, and sensing reveals that f is true, then we also have to specify that sensing f affects knowledge for all other possible values. Thus, for functional $f(\overline{x}, v)$ we add:

$$\begin{array}{l} v \neq v' \Rightarrow \\ \quad \textit{Initiates}\,\big(\textit{senseT}(f(\overline{x}, v)), KP(\neg f(\overline{x}, v')), t\big) \\ v \neq v' \Rightarrow \\ \quad \textit{Terminates}\,\big(\textit{senseT}(f(\overline{x}, v)), KP(f(\overline{x}, v')), t\big) \end{array} \tag{8}$$

We consider functional fluents to be only partial functions and therefore we do not add the respective axioms for *senseF*.

Goals can be specified as follows:

```
(:goal
 (and
  (or [3] inRoom(wc, livingRoom) )
  (or [4] !inRoom(vc, livingRoom) ) ) )
```

where the numbers in square brackets represents the time limit until the literal must be known to hold. Its general form is conjunctive normal form (CNF):

(and (or $[t_{1_1}]l_{1_1} \ldots [t_{1_n}]l_{1_n}$) ...
      (or $[t_{m_1}]l_{m_1} \ldots [t_{m_n}]l_{m_n}$))

It translates to epistemic EC as follows:

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} \textit{HoldsAt}\,\big(\textit{Knows}(l_{i_j}), t_{i_j}\big) \tag{9}$$

Initial Knowledge is specified as follows:

```
(:init
  inRoom(wc,corridor)
  inRoom(vc,corridor)
  isRobust(vc)
  hasDoor(corridor, d1)
  hasDoor(livingRoom, d1) )
```

It is a set $\mathcal{I}$ of literals which are known to hold at time 0. They translate to epistemic EC as:

$$\bigwedge_{l_i \in \mathcal{I}} \textit{HoldsAt}\,(\textit{KP}(l_i), 0) \tag{10}$$

EC does not use negation-as-failure, so we also have to explicitly state what the agent does not know:

$$\bigwedge_{l_i \in \mathcal{I}} l \neq l_i \Rightarrow \neg\textit{HoldsAt}\,(\textit{KP}(l), 0) \tag{11}$$

where the l_i are these literals the agent knows.

3.2 Action Specifications

Translating a non-epistemic action specification into an epistemic one is the trickiest part. Consider the action `moveRoomToRoom` for illustration:

```
(:action moveRoomToRoom
  :parameters (?robo - Robot ?door -
     Door ?from ?to - Room)
  :precondition
    (and  inRoom(?robo, ?from)
          !inRoom(?robo, ?to)
          hasDoor(?from, ?door)
          hasDoor(?to, ?door)
          (or isRobust(?robo)
              opened(?door)))
  :effect
    (if   (and  opened(?door)
                inRoom(?robo, ?from))
     then (and  inRoom(?robo, ?to)
               !inRoom(?robo, ?from))))
```

Informally it describes the conditional effect that if a robot executes this action it will end up in the target room if the door to the room is opened. The precondition says that the robot will only try to execute this action if it knows that the door is open or if it knows that it is robust (so crashing against a closed door does not cause harm). It consists of a parameters section, a precondition specification and a conjunction of conditional effects. The parameters are variables (denoted with the preceding `?`) which are universally quantified over the scope of their sort. An action's precondition is given in CNF:

(and (or $l_{1_1} \ldots l_{1_n}$) ... (or $l_{m_1} \ldots l_{m_n}$))

Let e be the action's name, then the the translated precondition is:

$$\textit{Happens}\,(e, t) \Rightarrow \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n} \textit{HoldsAt}\,\big(\textit{Knows}(l_{i_j}), t\big) \tag{12}$$

This states, that an event can only happen if its preconditions are known to hold. In terms of planning, it means that the planner will only consider actions in a plan when it *knows* that their preconditions hold.

A conditional effect of an action has the form:

(if (and $l_1 \ldots l_k$) then (and $l_1 \ldots l_m$))

It can be represented as a pair (E, C) where $C = \{l_1, \ldots, l_k\}$ is a set of condition literals and $E = \{l_1, \ldots, l_m\}$ is a set of effect literals. The translation of conditional effects into EEC is as follows:

First, even though we consider epistemic planning, the physical non-epistemic effects of actions are still valid. Thus, we have to add one non-epistemic effect axiom for each $l_j \in E$ to our theory:

$$\bigwedge_{l_i \in C} \mathit{HoldsAt}\,(l_i, t) \Rightarrow \pi(e, f_j, t) \tag{13}$$

where $l_i \in C$ are the condition literals, f_j are the fluents of the literals $l_j \in E$ and $\pi \in \{\mathit{Initiates}, \mathit{Terminates}\}$.

Second, considering epistemic effects of actions, we have to consider DECKT's axioms. They define that if all condition literals of a conditional effect are known to hold, then its effect literals are also known to hold. For every effect literal l_j in a conditional effect of an action we have to add:

$$\begin{aligned} &\bigwedge_{i=1}^{n} \mathit{HoldsAt}\,(\mathit{Knows}(l_i), t) \\ &\qquad\qquad \Rightarrow \mathit{Initiates}\,(e, \mathit{KP}(l_j), t) \end{aligned} \tag{14}$$

Knowledge about an effect fluent is lost if a) at least one of the conditions is unknown and if b) there is no condition which is known not to hold and if c) the new truth value of the effect fluent is not already known. Thus, for every $l_j \in E$ we have to add the following effect axiom:

$$\begin{aligned} &\left(\bigvee_{l_i \in C} \neg \mathit{HoldsAt}\,(\mathit{Knows}(l_i), t) \right) \wedge \\ &\left(\bigwedge_{l_i \in C} \neg \mathit{HoldsAt}\,(\mathit{Knows}(\neg l_i), t) \right) \wedge \\ &\neg \mathit{HoldsAt}\,(\mathit{Knows}(l_j), t) \\ \Rightarrow\; &\mathit{Terminates}\,(e, \mathit{KP}(l_j), t) \end{aligned} \tag{15}$$

DECKT defines how *knowledge about the effect of an action is obtained through knowledge about its condition*. This is useful, e.g. if for some reason sensing a door's open-state is not possible in certain rooms. In this case one can send the robot through the door, sense its open-state (condition) *after* the execution and *retroactively* infer the robot's location (effect).

If a conditional effect has more than one condition and if we want to be complete wrt. to DECKT, we have to consider every subset of the conditions to be potentially unknown, i.e. if C is the set of literals in a condition, we have to consider every non-empty set $C_u = \{C_i \in 2^C | C_i \neq \emptyset\}$. If this subset is unknown at the action's execution time and all other conditions $C_k = C \backslash C_u$ are known to hold, then the effect is also unknown at execution time (see Equation 15). However, if knowledge about the unknown conditions is acquired later (e.g. through sensing) and the conditions are revealed to hold, then the effect is also known to hold. Further, the effect is known not to hold if the conditions are revealed not to hold. At this stage of our work we do not aim for completeness wrt. to DECKT and only regard the case where one condition is unknown. This also reduces complexity because we do not have to consider every element of the power set 2^C to be potentially unknown but only every single element of C.

Thus, for every effect literal $l_j \in E$ and for every potentially unknown condition literal $l_u \in C$ we have to add the following sentences to the theory:

$$\begin{aligned} &\left(\bigwedge_{l_i \in C \backslash \{l_u\}} \mathit{HoldsAt}\,(\mathit{Knows}(l_i), t) \right) \wedge \\ &\neg \mathit{HoldsAt}\,(\mathit{Knows}(\neg l_u), t) \wedge \\ &\neg \mathit{HoldsAt}\,(\mathit{Knows}(l_j), t) \\ \Rightarrow\; &\mathit{Initiates}\,(e, \mathit{KP}(l_u \Rightarrow l_j), t) \end{aligned} \tag{16}$$

The event initiates an *implication*, stating that if l_u is true then l_j must also be true.

In order to *acquire knowledge about a condition through an effect*, the effect of an action must be known not to hold at the action's execution time, and there must be no condition which is already known not to hold. If the effect later becomes known to hold, then causality tells us that all conditions of the actions must also hold. Thus, for each condition literal $l_j \in C$ and each effect literal $l_i \in E$, we add the following sentence to our theory:

$$\begin{aligned} &\neg \mathit{HoldsAt}\,(\mathit{Knows}(l_i), t) \wedge \\ &\neg \mathit{HoldsAt}\,(\neg \mathit{Knows}(l_j), t) \wedge \\ &\neg \mathit{HoldsAt}\,(\mathit{Knows}(l_j), t) \\ \Rightarrow\; &\mathit{Initiates}\,(e, \mathit{KP}(l_j \Rightarrow l_i), t) \end{aligned} \tag{17}$$

DECKT uses the predicate *KmAffect* to express that an event may affect a fluent. The predicate holds for each effect $l_j \in E$ of a conditional effect (C, E) of an event e if there is no condition which is not known not to hold:

$$\begin{aligned} &\mathit{Happens}\,(e, t) \wedge \bigwedge_{l_i \in C} \neg \mathit{HoldsAt}\,(\mathit{Knows}(\neg l_i), t) \\ &\Rightarrow \mathit{KmAffect}(e, f_j, t) \end{aligned} \tag{18}$$

where f_j is the fluent in the literal l_j.

KmAffect is used to trigger the termination of implications. That is, an implication is terminated if one of the involved fluents may be affected.

$$\begin{aligned} &\mathit{HoldsAt}\,(\mathit{KP}(l \Rightarrow l'), t) \wedge \\ &\mathit{KmAffect}(e, f, t) \vee \mathit{KmAffect}(e, f', t) \\ \Rightarrow\; &\mathit{Terminates}\,(e, \mathit{KP}(l \Rightarrow l'), t) \end{aligned} \tag{19}$$

There are two more axioms required for soundness and completeness wrt. DECKT which we do not explicitly write down in this paper for brevity reasons. The first one handles how knowledge which is gained through an implication becomes persistent if the implication is terminated. The second one handles transitivity of implications which sometimes must be made explicit. We point the interested reader to (Patkos, 2010) and refer to these two axioms as (RT).

Observations are performed by actively using sensors. They are a key element of epistemic theories, as they provide the agent's basis for knowledge acquisition. In PLIK we use the keyword `:observation` to describe sensing effects. Consider the following example:

```
(:action senseInRoom
  :parameters ( ?robo - Robot
                ?room - Room  )
  :precondition
  :effect
  :observation
    (and  inRoom(?robo, ?room) ) )
```

This action does not have a precondition or a physical effect and is thus a pure sensing action.[6] The observations are provided as a conjunction `(and` $f_1 \dots f_{|\mathcal{O}|}$`)`.

Let e be the action's name and $\overline{v}$ universally quantified variables according to the action's parameters, then for each observed f_i we add the following disjunctive event axiom:

$$\begin{aligned}&\mathit{Happens}\,(e(\overline{v}),t) \Rightarrow &(20)\\ &\mathit{Happens}\,(senseT(f_i),t) \vee \mathit{Happens}\,(senseF(f_i),t)\end{aligned}$$

with $f_i \in \overline{v}$. Intuitively, this means if a sensing action happens, then either positive or negative sensing will happen.

3.3 Solving Epistemic Planning Problems

Given a planning problem description in PLIK, we generate an EEC representation as follows:

1. Objects, types and fluents are declared.
2. For each fluent we declare and specify *senseT* and *senseF* events Σ_s as described in (7) and (8). For functional fluents we also apply (6), which results in a set of constraints Ψ.
3. Initial knowledge Γ is specified as described in (10), (11) and goals Γ' are specified as described in (9).
4. Action specifications Σ are generated through (12), (13), (14), (15), (16), (17), (18) and (20).

[6]However, note that in principle PLIK and EEC allow to define actions which have both a physical effect and an observation.

The pair (Δ, Γ^*) is a *possible solution* to the planning problem with incomplete knowledge if the following holds:

$$\begin{aligned}&\mathit{CIRC}[\Sigma \wedge \Sigma_s; \mathit{Initiates, Terminates, Releases}] \wedge &(21)\\ &\mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma \wedge \Gamma^* \wedge \Omega \wedge \mathit{EC} \wedge \mathit{DECKT}\\ &\models \Gamma'\end{aligned}$$

where $\mathit{DECKT} = (2) \wedge (3) \wedge (4) \wedge (5) \wedge (19) \wedge (RT)$ represents the domain-independent DECKT axioms which are not explicitly encoded in the theory through our translation. Γ^* can be interpreted as a *possible world* in which the plan Δ is a solution to the planning problem, i.e. Γ^* is a conjunction of *HoldsAt* statements which can be seen as a condition under which Δ solves the planning problem. Figure 2 shows how Γ^* and Δ are included in the output stable models of the ASP tools.

The complete solution of the planning problem is the set of all n pairs (Δ_n, Γ_n^*) for which the entailment (21) holds. This solution can be interpreted as a conditional plan with the conditions represented by Γ_i^* and the actions to be executed under a certain condition by Δ_i.

The execution of this plan also demands for an execution monitor which compares the possible worlds Γ_i^* with the agent's knowledge about the world. At execution time, the agent non-deterministically chooses a plan Δ_i and follows it as long as the condition Γ_i^* is not inconsistent with the agent's knowledge about the world. If the Γ_i^* becomes inconsistent due to sensing actions, then the agent has to choose another branch (Δ_j, Γ_j^*) with a Γ_j^* that is consistent with the agent's knowledge about the world. If no such Γ_j^* exists, then the problem is unsolvable.

3.4 Implementation Issues

Lee and Palla (2012) propose to use Answer Set Programming to solve EC reasoning problems and show that their approach outperforms existing solutions like the DEC reasoner by Mueller (2005). We make use of their f2lp tool to translate the EEC planning problem description into an ASP representation of the problem. We use the tool gringo (Gebser et al., 2011) to generate the Herbrand Models of the problem and claspD (Drescher et al., 2008) as the ASP solver.

Unfortunately, not the f2lp-tool we use nor any other reasoner we know about supports reification. Therefore we had to sacrifice completeness of our planner and define special predicates which translate as follows:

$$\begin{aligned}
&\textit{Knows}(f,t) := \textit{HoldsAt}\,(\textit{Knows}(f),t)\\
&\textit{KnowsNot}(f,t) := \textit{HoldsAt}\,(\textit{Knows}(\neg f),t)\\
&\textit{KP}(f,t) := \textit{HoldsAt}\,(\textit{KP}(f),t)\\
&\textit{KPNot}(f,t) := \textit{HoldsAt}\,(\textit{KP}(\neg f),t)\\
&\textit{ImpliesTT}(f,f',t) := \textit{HoldsAt}\,\big(\textit{KP}(f \Rightarrow f'),t\big)\\
&\textit{ImpliesTF}(f,f',t) := \textit{HoldsAt}\,\big(\textit{KP}(f \Rightarrow \neg f'),t\big)\\
&\textit{ImpliesFT}(f,f',t) := \textit{HoldsAt}\,\big(\textit{KP}(\neg f \Rightarrow f'),t\big)\\
&\textit{ImpliesFF}(f,f',t) := \textit{HoldsAt}\,\big(\textit{KP}(\neg f \Rightarrow \neg f'),t\big)
\end{aligned} \tag{22}$$

We define the persistence laws for the *KP* and *Implies* predicates analogously to the persistence laws for the *HoldsAt* predicate in the DEC axiomatization and use special *Initiates* and *Terminates* predicates for each of the *KP* and *Implies* which follow the very same axioms as the original DEC *Initiates* and *Terminates* do. For example, we have

$$\begin{aligned}
&\textit{InitiatesImpliesTF}(e,f,f',t) :=\\
&\qquad\qquad \textit{Initiates}\,(e,\textit{KP}(f \Rightarrow \neg f'),t)
\end{aligned}$$

and our theory contains the axioms

$$\begin{aligned}
&\textit{Happens}\,(e,t) \wedge \textit{InitiatesImpliesTF}(e,f,f',t)\\
\Rightarrow\ &\textit{ImpliesTF}(f,f',t+1)\\
&\textit{ImpliesTF}(f,f',t) \wedge\\
&\neg\exists e : (\textit{Happens}\,(e,f,t) \wedge\\
&\textit{TerminatesImpliesTF}(e,f,f',t))\\
\Rightarrow\ &\textit{ImpliesTF}(f,f',t+1)
\end{aligned}$$

which are equivalent to the usual DEC axioms for *HoldsAt* by Mueller (2005).[7] Contraposition and transitivity rules for the implication predicates is also implemented manually.

Using these special predicates and the additional persistence and effect axioms for each predicate, our approach is sound. This is easy to see as none of our translation rules generates an implication which involves more than two fluents. However, it is not complete as we do not include HCD expansion as described in Patkos (2010). Further, we do not consider the gain of knowledge about effects if more than one condition is unknown (see Equation (16)).

4 EVALUATION

Based on a prototypical implementation we generate formalizations based on problems already mentioned in literature and our running example. We investigate the reasoning results and provide another result concerning usability, i.e. the reduction of the planning problem specification.

4.1 Reduction of the planning problem specification

Looking at the translation of PLIK to non-epistemic EC, we find that through (13), for each conditional effect (C, E), $|E|$ effect axioms of the form $\gamma \Rightarrow \pi(e, f, t)$ are generated; one for each effect literal $l_j \in E$. Now, looking at the translation into the epistemic EC, we find that many additional epistemic effect axioms are generated: (14) generates $|E|$ effect axioms, (15) generates another $|E|$ axioms and (16), (17) generate $|E| \cdot |C|$ axioms each. Thus, each conditional effect (C, E) of an action specification demands specifying $|E|$ non-epistemic effect axioms and $|E| \cdot (2 \cdot |C| + 2)$ epistemic effect axioms.

For example, consider extending the `moveRoomToRoom` action by adding only two more conditions (robot's battery must be full and robot must not be blocked) to the effect:

```
(:action moveRoomToRoom
  :parameters (?robo - Robot ?door -
     Door ?from ?to - Room)
  :precondition
    (and  inRoom(?robo, ?from)
          !inRoom(?robo, ?to)
          hasDoor(?from, ?door)
          hasDoor(?to, ?door)
          (or isRobust(?robo)
              opened(?door)))
  :effect
    (if   (and  opened(?door)
                batteryFull(?robo)
                !blocked(?robo)
                inRoom(?robo, ?from))
     then (and  inRoom(?robo, ?to)
               !inRoom(?robo, ?from))))
```

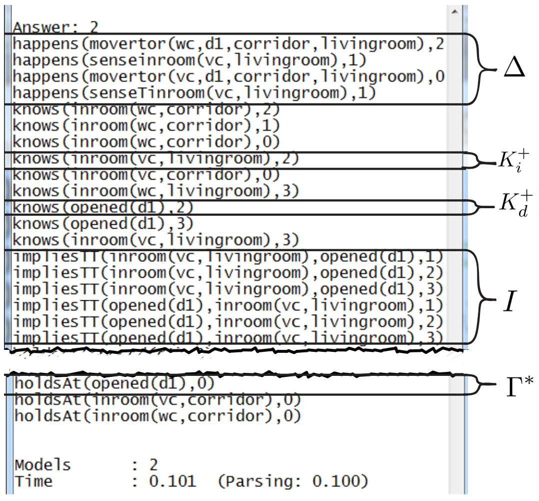
Then, considering all knowledge-level effects of this simple action with 4 conditions, the epistemic EC version of the action is more than 100 lines (11 000 bytes) of axioms which would be very circumstantial to implement manually, especially when considering epistemic accuracy of the axioms.

4.2 Use cases

To demonstrate applicability of our approach we implement the running example of the robot moving through a door and implement another use case from literature (Petrick and Bacchus, 2004).

Move-through-door example: We implemented a use case with several robots moving through rooms and one central planning agent controlling the robots.

[7] We do not have to account for *Release* axioms, as the *Implies* and *KP* predicates are always subject to inertia.



```
Answer: 2
happens(movertor(wc,d1,corridor,livingroom),2
happens(senseinroom(vc,livingroom),1)
happens(movertor(vc,d1,corridor,livingroom),0
happens(senseTinroom(vc,livingroom),1)
knows(inroom(wc,corridor),2)
knows(inroom(wc,corridor),1)
knows(inroom(wc,corridor),0)
knows(inroom(vc,livingroom),2)
knows(inroom(vc,corridor),0)
knows(inroom(wc,livingroom),3)
knows(opened(d1),2)
knows(opened(d1),3)
knows(inroom(vc,livingroom),3)
impliesTT(inroom(vc,livingroom),opened(d1),1)
impliesTT(inroom(vc,livingroom),opened(d1),2)
impliesTT(inroom(vc,livingroom),opened(d1),3)
impliesTT(opened(d1),inroom(vc,livingroom),1)
impliesTT(opened(d1),inroom(vc,livingroom),2)
impliesTT(opened(d1),inroom(vc,livingroom),3)

holdsAt(opened(d1),0)
holdsAt(inroom(vc,corridor),0)
holdsAt(inroom(wc,corridor),0)


Models      : 2
Time        : 0.101   (Parsing: 0.100)
```

Figure 2: The output of the reasoner for the move-through-door example with two robots (vc,wc)

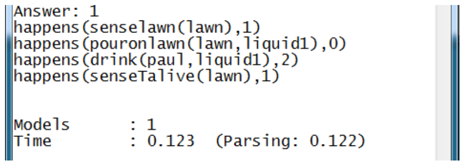
Some of the robots are equipped with bumpers, so they are robust and it is safe to send them through a door without knowing whether the door is open. In the worst case the robots hit the door but they don't break. Robots which don't have a bumper break if they hit a closed door. For safety, the planning agent only considers sending them through doors if it knows that the doors are open.[8]

We investigated the stable models which are generated by the reasoner and it turned out that the knowledge-level effects of actions are correctly handeled. Figure 2 shows how implications (I), plan (Δ), assumptions about the world (Γ^*), direct knowledge gain (K_d^+) as well as indirect knowledge gain (K_i^+) are successfully generated and modeled. The output is that of a simple scenario with two robots, `wc` and `vc`, where `vc` is robust and `wc` is not. The generated plan involves sending the robust `vc` through a door at $t = 0$, sensing its location at $t = 1$ and then, knowing that the door must be open at $t = 2$, safely sending `wc` through the door.

Poisonous liquid We adopted the poisonous liquid-example from Petrick and Bacchus (2004). A thirsty agent has a liquid and does not know whether it is poisonous. He can perform the following drink-action:

```
(:action drink
  :parameters (?li - Liquid ?a - Agent)
  :precondition
  :effect
    (if poisonous(?li)
    then poisoned(?p))
    !thirsty(?p))
```

[8]See the corresponding move-action in section 3.2.



```
Answer: 1
happens(senselawn(lawn),1)
happens(pouronlawn(lawn,liquid1),0)
happens(drink(paul,liquid1),2)
happens(senseTalive(lawn),1)


Models      : 1
Time        : 0.123   (Parsing: 0.122)
```

Figure 3: The resulting output for the poisonous liquid example

If he drinks the liquid he will not be thirsty anymore but he may get poisoned if the liquid is poisonous.

The agent can also pour the liquid on the lawn. If the liquid is poisonous the lawn will not be alive anymore:

```
(:action pourOnLawn
  :parameters (?li - Liquid ?la - Lawn)
  :precondition
  :effect
    (if poisonous(?li)
    then !alive(?la)))
```

Finally the agent can sense whether the lawn is alive or not:

```
(:action senseLawn
  :parameters (?la - Lawn)
  :precondition
  :observation
      alive(?la))
```

In the initial state, the agent only knows that he is thirsty and not poisoned. The goal is that the agent always knows that he is not poisoned and that he knows that in the end he is not thirsty. The result is exactly 1 stable model, because there is only one possible world in which the goal can be achieved. This is the world where the liquid is not poisonous. We show the plan, i.e. the actual happens statements of the stable model in Figure 3. For sake of brevity, the figure only shows a subset of the stable model, i.e. only the *happens* statements. The agent `paul` first pours the liquid on the lawn, then senses whether the lawn is dead and finally, if the lawn is not dead, drinks the liquid. Note that the implicit condition $\Gamma^* = \neg \mathit{HoldsAt}(\mathit{poisonous(liquid1)}, 0)$ is not part of the output stable model, because in terms of stable models $\neg$ is to be interpreted as weak negation and a stable model by definition does not contain weakly negated literals.

5 CONCLUSION

In this paper we provide a method for the formalization of epistemic planning problems. The main contribution of our approach lies in the automated translation of planning problem specifications which assume complete knowledge about the world into planning problems which allow for incomplete knowledge. We show that this translation safes a problem designer the work of specifying additional $|E| \cdot (2 \cdot |C| + 2)$ knowledge-level effect axioms which also must be epistemically accurate. This result proves our hypothesis that the planning problem representation is drastically simplified with our approach.

To the best of our knowledge there is currently no other planning system which takes ordinary planning domains as input and automatically compiles them into epistemic planning domains, such that epistemic effects of actions can be exploited. The work by Sardina et al. (2004) regards epistemically accurate theories, but apart from that no planner we know about exists which performs a soundness-check on given epistemic action specifications.

We present use cases which illustrate how sensors can be used in a more flexible way, i.e. how sensors can replace other sensors when using our EP approach. Thus, our second hypothesis is also affirmed.

Our approach is sound but not complete wrt. DECKT and the possible worlds semantics of knowledge. To achieve completeness we have to introduce actions with non-deterministic effects, e.g. like tossing a coin. Another issue that we need to consider is knowledge acquisition about effects if more than one condition is unknown. This goes along with what Patkos (2010) calls HCD-expansion. However, this is hardly possible without true reification and we don't know of any reasoner which supports this. Nevertheless, if true reification would be applied, complexity is increased significantly.

Future research also includes the application in a real robotic domain. In particular, we are currently implementing an execution monitor, so the conditional plans provided by the reasoner can be executed.

6 ACKNOWLEDGEMENTS

We are very grateful for the help from Theodore Patkos who was always happy to answer questions and to help us with details concerning DECKT and related work.

Funding by the German Research Association (DFG) under the grants of International Research Training Group on Semantic Integration of Geospatial Information (IRTG SIGI) and SFB/TR8 Spatial Cognition is gratefully acknowledged.

REFERENCES

Bertoli, P., Cimatti, A., Lago, U. D., and Pistore, M. (2002). Extending PDDL to nondeterminism, limited sensing and iterative conditional plans. In *ICAPS Workshop on PDDL*.

Bertoli, P., Cimatti, A., Pistore, M., Roveri, M., and Traverso, P. (2001). MBP : a Model Based Planner. In *IJCAI Proceedings*.

Blackburn, P., de Rijke, M., and Venema, Y. (2001). *Modal Logic*. Cambridge University Press.

Bolander, T. and Birkegaard Andersen, M. (2011). Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–33.

Brenner, M. and Nebel, B. (2009). Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331.

De Giacomo, G. and Levesque, H. J. (1998). An incremental interpreter for high-level programs with sensing. Technical report, Department of Computer Science, University of Toronto.

Doherty, P., Gustafsson, J., Karlsson, L., and Kvarnström, J. (1998). Temporal Action Logics (TAL): Language specification and tutorial. *Computer and Information Science*, 3(015).

Drescher, C., Gebser, M., Grote, T., Kaufmann, B., König, A., Ostrowski, M., and Schaub, T. (2008). Conflict-Driven Disjunctive Answer Set Solving. In *International Conference on Principles of Knowledge Representation and Reasoning*.

Eiter, T., Faber, W., Leone, N., Pfeifer, G., and Polleres, A. (2000). Planning under Incomplete Knowledge. In *International Conference on Computational Logic*.

Gebser, M., Kaminski, R., König, A., and Schaub, T. (2011). Advances in gringo series 3. In *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning*.

Gelfond, M. and Lifschitz, V. (1988). The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference on Logic Programming (ICLP)*.

Hoffmann, J. and Brafman, R. (2005). Contingent planning via heuristic forward search with implicit belief states. In *ICAPS Proceedings*, volume 2005.

Kowalski, R. (1986). A Logic-based calculus of events. *New generation computing*, 4:67–94.

Kushmerick, N., Hanks, S., and Weld, D. S. (1995). An algorithm for probabilistic planning. *Artificial Intelligence*, 76.

Kvarnström, J. (2005). *TALplanner and other extensions to temporal action logic*. PhD thesis, Lingköpings Universitet.

Lee, J. and Palla, R. (2009). System f2lp Computing Answer Sets of First-Order Formulas. In *Logic Programming and Nonmonotonic Reasoning*, pages 515–521.

Lee, J. and Palla, R. (2012). Reformulating the Situation Calculus and the Event Calculus in the General Theory of Stable Models and in Answer Set Programming. *Journal of Artificial Intelligence Research*, 43:571–620.

Mcdermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., and Wilkins, D. (1998). PDDL The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control.

Moore, R. (1985). A formal theory of knowledge and action. In Hobbs, J. and Moore, R. C., editors, *Formal theories of the commonsense world*. Ablex, Norwood, NJ.

Mueller, E. (2005). *Commonsense reasoning*. Morgan Kaufmann.

Patkos, T. (2010). *A Formal Theory for Reasoning About Action , Knowledge and Time*. PhD thesis, University of Crete - Heraklion Greece.

Patkos, T. and Plexousakis, D. (2009). Reasoning with Knowledge , Action and Time in Dynamic and Uncertain Domains. In *IJCAI Proceedings*, pages 885–890.

Petrick, R. P. A. and Bacchus, F. (2004). Extending the knowledge-based approach to planning with incomplete information and sensing. In *ICAPS Proceedings*.

Pryor, L. and Collins, G. (1996). Planning for Contingencies : A Decision-based Approach. *Journal of Artificial Intelligence Research*, pages 287–339.

Reiter, R. (2001). *Knowledge in action: Logical foundations for specifying and implementing dynamical systems*. MIT Press.

Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Sardina, S., Giacomo, G. D., Lespérance, Y., and Levesque, H. J. (2004). On the semantics of deliberation in IndiGolog from theory to implementation. *Annals of Mathematics and Artificial Intelligence*, pages 259–299.

Scherl, R. and Levesque, H. J. (2003). Knowledge, action, and the frame problem. *Artificial Intelligence*.

Shanahan, M. (2000). An abductive event calculus planner. *The Journal of Logic Programming*, pages 207–240.

Thielscher, M. (1998). Introduction To The Fluent Calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(14).

Thielscher, M. (2005). FLUX : A Logic Programming Method for Reasoning Agents. *Theory and Practice of Logic Programming*, 5(4-5).

van Ditmarsch, H., van der Hoek, W., and Kooi, B. (2007). *Dynamic Epistemic Logic*. Springer.