Epistemic Planning and Games

Helge Hatteland, s144457 Thomas Pethick, s144448 Oliver Fleckenstein, s144472

***Abstract*—This paper presents a summary of the articles [12][8], describing how a dynamic epistemic logic framework can be used for epistemic planning in games. First of all, we introduce important notation and concepts of dynamic epistemic logic. Secondly, an implementation of a minimalistic example of the game *Thief: The Dark Project*^{TM} is presented. Finally, we utilize game theory to devise optimal strategies, considering different outcomes based on randomly occurring events.**

I. Introduction

We show how dynamic epistemic logic can be used to model agents in a computer game to attain more human like behaviour. This is done by presenting the work and toy example of [12] followed by an implementation in AgentSpeak. A more accessible introduction to epistemic logic is included and the behaviour of the event models is covered in more detail. Using elements from game theory, we analyze the agents' information and options in the game, devising a strategy for the agents to reach the optimal outcome.

A. Outline of this paper

In § II the necessary notation and concepts for dynamic epistemic logic (DEL) is introduced. It includes a definition of epistemic logic which is then extended to DEL in § III by describing the action model of which three specific instances are covered: secret message, private messages and public announcements. We then proceed in § IV to model a toy example inspired by the computer game Thief with higher-order knowledge possible to capture in DEL. We include an implementation in AgentSpeak and discuss the limitations compare to the DEL powered knowledge module. Finally in § V a game theoretical interpretation of scenario is considered that discusses the strategy developed in § IV. This provides a framework for modelling the probabilistic behaviour of *nature* and analyze a sequence of actions. In § VI we close with pointers to modelling belief and planning longer sequences.

II. Epistemic Logic

Epistemic Logic is the logic for reasoning about knowledge. It concerns itself with the knowledge of multiple agents in a system where agent should be understood in a broad sense to include not only people but also robots and even components in a computer system. It models agents knowledge about facts as well as knowledge about other agents knowledge referred to as *high-order* knowledge. The concepts will be introduced using a combination of the notation used by [9] and [6].

A. Epistemic Model

The key idea behind epistemic logic is that of *possible worlds*. An agent is informed of φ if it is true in the range of possible worlds *accessible* from the current world. To make these notions precise an epistemic model is defined.

Definition II.1. Epistemic model (possible world model) An epistemic model M for n agents over a set of atomic propositions Φ is a tuple $M = (S, \pi, \mathcal{K}_1, ..., \mathcal{K}_n)$ satisfying,

1) S is a non-empty set of states (i.e. worlds).
2) π which for every state associates a truth assignment to the propositions, $s \in S, \pi(s) : \Phi \to \{ true, false \}$.
3) $\mathcal{K}_i$ is a binary relation on S for every $i \in \mathcal{A}$ which is called an *accessibility relation*.

The model is used to depict what worlds are accessible and what holds in those worlds. It is however beneficial to consider a specific world and use the accessibility relations from this. This specific *scenario* in our model will be referred to as a Pointed possible world model.

Definition II.2. Pointed possible world model A pointed possible world model is a pair (M, s) where $M = (S, \pi, \mathcal{K}_1, ..., \mathcal{K}_n)$ and $s \in S$.

This model can capture the knowledge of the agents in the modelled situation. To express more clearly what is known a propositional modal language $\mathcal{L}_{EL}$ is introduced as defined by the following grammar.

Definition II.3. Language $\mathcal{L}_{EL}$

$$\mathcal{L}_{EL} : \varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid K_i\varphi,$$

where $\top$ is a tautology, p is a atomic proposition $p \in \Phi$, $\mathcal{A} = \{ 1, ..., n \}$ is a set of agents, and $i \in \mathcal{A}$.

This can simply be seen as an extension to propositional logic with the additional modal operator $K_i\varphi$ which reads as *agent i knows that* φ. The rest of the connectives ($\vee, \to, \leftrightarrow, \oplus, \uparrow, \downarrow$) can be expressed based on negation and logical and as usual [1, Theorem 2.36].

This language can be used to describe what is true in a particular scenario. If $\varphi \in \mathcal{L}_{EL}$ then we express that φ holds in the world s by $(M, s) \vDash \varphi$. The semantics of the language makes this precise.

Definition II.4. Semantics for $\mathcal{L}_{EL}$.

$(M, s) \vDash \top$	
$(M, s) \vDash p$	iff $\pi(s)(p) = true$
$(M, s) \vDash \neg\varphi$	iff $(M, s) \nvDash \varphi$
$(M, s) \vDash \varphi \wedge \psi$	iff $(M, s) \vDash \varphi$ and $(M, s) \vDash \psi$
$(M, s) \vDash K_i\varphi$	iff for all v with $(s, v) \in \mathcal{K}_i, (M, v) \vDash \varphi$

Definitions for concepts such as *common knowledge* are left out for simplicity since they can be defined in terms of the modal operator. To provide an intuition of these semantics an interpretation of the semantics in natural language is helpful.

1) $K_i\varphi$: agent i knows φ.
2) $\neg K_i\varphi$: agent i does not know φ.
3) $\neg K_i\neg\varphi$: agent i considers φ possible.
4) $\neg(K_i\varphi \wedge K_i\neg\varphi)$: agent i does not know whether φ or not φ.

B. Kripke Structure

The epistemic model is what is called a Kripke structure in model checking which has a useful graphical representation. Each state is a vertice in a graph connected by edges which are the accessibility relations. In our description of the model vertices, state and world will be used interchangeably. The label of state $s \in S$ describe what truth assignments the atomic propositions has in that state. In a pointed world model the pointed vertice will be indicated by a double border of which an example can be found on figure 1a.

Note that it is common to assume that the accessibility relation is an *equivalence* relation. That is the relation is,

- *reflexive*, which is that if $s \in S$ then $(s, s) \in \mathcal{K}$,
- *symmetric*, which means that for all $s, t \in S$ we have $(s, t) \in \mathcal{K}$ iff $(t, s) \in \mathcal{K}$, and
- *transitive*, which means that for all $s, t, u \in S$ if $(s, t) \in \mathcal{K}$ and $(t, u) \in \mathcal{K}$ then $(s, u) \in \mathcal{K}$.

This means that the graphical representation used will leave out edges implicit by this assumption. However, we will not assume reflexivity since leaving out this property will be useful will constructing *actions* later introduced. For consistency reflexive relations will also be made explicit in epistemic models.

III. Dynamic Epistemic Logic

Dynamic Epistemic Logic (DEL) provides a well-defined way of updating the epistemic model based on so called event models. It consists of three parts of which the first have already been covered: 1) an initial *epistemic model* 2) an *event model* and 3) a *product update* operator. It is based on the work by [8] but adopts the style of notation from [6].

A. Event Model

An event model $\mathcal{E}$ is a tuple $\mathcal{E} = (E, Pre, R_i, ..., R_n)$ where E is the set of events, $R_i \subset E \times E$ is an accessibility relation for each agent $i \in \mathcal{A}$, and $Pre : S \to \mathcal{L}_{EL}$ is the precondition.

Note how it is structurally similar to an epistemic model. The difference will show in how it is applied in the product update.

B. Product Update

Product update is an operator that produces a new epistemic model based on an event occurring in an initial epistemic model. Formally, given an epistemic model, $M = (S, \pi, \mathcal{K}_1, ..., \mathcal{K}_n)$, and event model, $\mathcal{E} = (E, Pre, R_i, ..., R_n)$, a new model is defined as,

$$M \otimes \mathcal{E} = (S', \pi', \mathcal{K}'_1, ..., \mathcal{K}'_n)$$

where,

1) $S' = \{\, (s, e) \mid s \in S, e \in E, (M, s) \vDash Pre(e) \,\}$,
2) $\pi'((s, e)) = \pi(s)$ and
3) $((s, e), (s', e')) \in R_i$ iff $(s, s') \in \mathcal{K}_i$ and $(e, e') \in R_i$ for $i \in \mathcal{A}$.

Informally this reads that a world (s_i, e_j) is only persisted in which the precondition is met. Further, an accessibility relation only exists if it also existed in both the initial epistemic model and event model.

To obtain the language for dynamic epistemic logic we extend $\mathcal{L}_{EL}$ with a modal operator $\langle \mathcal{E}, e \rangle$. The semantics for these modalities are then,

$$(M, s) \vDash \langle \mathcal{E}, e \rangle \varphi \text{ iff}$$
$$(M, s) \vDash Pre(e) \text{ and } (M \otimes \mathcal{E}, (s, e)) \vDash \varphi$$

C. Locked Box Example

Let us consider an example to get a better intuition of how to design the event model and how the product update works. Suppose a box exists with two possible combination to open it, c_1 and c_2. In our model two agents exists, i and j, where i is the only one that knows the combination which is c_1. We can describe this static situation with the epistemic model expressed by the Kripke model in figure 1a.

We can verify that it has the desired properties by evaluating different formulas. First it is clear that i knows the combination for any pointed world model since there is no accessibility relation between the two states. However, because $(s_1, s_3) \in \mathcal{K}_j$ j will always consider both a world in which c_1 and one in which c_2.

The formalization has clarified certain points that were ambiguous in the natural language formulation. It is explicit in the model that j knows that i knows the combination (formally $K_j(K_ic_1 \vee K_ic_2)$). The interpretation also includes the assumption that only one combination exists. In other words that no world exists in which both $\pi(c_1)$ and $\pi(c_2)$ are either true or false. These two worlds are left out for simplicity since they have no accessibility relations with the other worlds and are not the pointed world considered. It would be possible to capture the model with one propositional atom where the negation would indicate that the second key was the case. However, this provides a slightly more complicated example to express the graphical representation.

How this model can be updated will now be addressed. This is done by constructing event models which leads to specific outcomes by application of the product update.

1) Secret message: A useful notion is that of a *secret message* which informally is an event in which an agents learns a fact without other agents being informed. In the context of the locked box example say j learns the combination without agent i's knowledge. To be precise, this implies that $K_j(c_1 \wedge \neg c_2)$ and $\neg K_i(\neg K_j(c_1 \wedge \neg c_2))$ should hold in the resulting pointed world model. An event model that contains this event is captured in figure 1b which reasoning behind will now be explained. First, note that the precondition in the pointed event e_1 matches the interpretation of c_1 in the epistemic model. No accessibility relations exists for j from

this state except the reflexive relation. This captures that j will considers no other world than the ones in which $c_1 \wedge \neg c_2$. In contrast i only has relations to all other states (in this case e_2). Since the precondition $Pre(e_2)$ is always true (expressed by $\top$) i will consider all worlds possible in the updated model for which i also considered them possible in the initial epistemic model. Finally note that e_2 expresses the event in which j was not informed. In this case j knows that j was not informed which is captured by the lack of an accessibility relation to e_1. However, if e_2 occurred agent i still considers worlds in which e_1 occurred (i.e. let S' be the all worlds consider in S then $S' \times E$ are considered).

This is an informal explanation that only provides a hand-wavy reasoning behind the event model. To ensure that it behaves as expected the resulting model $M \otimes \mathcal{E}$ should be constructed and the desired formulas evaluated in the pointed model $(M \otimes \mathcal{E}, (s_1, e_1))$ which can be found on figure 1c. First note that the property of i's certainty of the state either consisting of s_1 or s_2 is preserved as is clear from the lack of accessibility relations. Secondly the sub graph containing vertices (s_1, e_2) and (s_2, e_2) is isomorphic to the initial Kripke structure M (made explicit in figure 1c by coloring). So the updated structure models the same situation as before if e_1 is not in the pointed world. Now consider the remaining state (s_1, e_1) which is the most interesting since this is the actual scenario, i.e. from this state our initial requirements for the event should hold by applying the semantics. The agent j only has a reflexive connection from (s_1, e_1) so he correctly only considers a world in which $c_1 \wedge \neg c_2$. Verifying i's knowledge similarly $((s_1, e_1), (s_1, e_2)) \in \mathcal{K}_i'$ captures both knowledge of the event is unknown to i by only considering (s_1, e_2) and that i knows $c_1 \wedge \neg c_2$.

2) Private message: The structure of a private message is very similar to that of a secret message. Private messages differs only by additionally capturing that other agents knows the recipient has received a message. This is formalized by removing (e_2, e_2) from the accessibility relation of i in the secret message event model as exemplified by figure 2. Expressing the situation in the modal language initially if j does not know about φ then $K_i(\neg K_j \varphi \wedge \neg K_j \neg \varphi)$. However, after the message has been parsed to j, i is also informed about the knowledge acquisition, so $K_i(K_j \varphi \vee K_j \neg \varphi)$ holds in pointed model.

3) Public announcement: As a last example it is shown how to model the concept of public announcement where every agent acquires a fact. The simple event model can be found on figure 3. It simply works by effectively filtering out worlds in which the precondition for e_1 is not met but leaving all accessibility relations.

4) Other messages: A more complete picture of possible standard messages including e.g. group messages can be found in [4][5]. This includes the generalization of the covered messages.

All knowledge acquisition has so far considered an all-knowing oracle, i.e. the sender is 'outside' the model. For message parsing that includes agency we refer to [10] which uses a history-based approaches to also include past events. DEL is sufficient for capturing the scenarios discussed in this paper.

D. Limitations of Dynamic Epistemic Logic

As seen dynamic epistemic logic allows one to update the knowledge about facts and knowledge about knowledge in a model. It does not however allow for changes to the facts themselves (i.e. the interpretation $\pi(s)$ in each state s). This is a possible extension to the product update that is only left out for simplicity.

A more serious concern is that the updated model after a secret message cannot be further updated in any meaningful way to inform i that j knows c_1. To manage this we could model beliefs which can be false compared to knowledge. This can be done with dynamic doxastic logic but DEL works for the toy-examples discussed.

This event based approach will allow us to model the possible sequences of actions in a game as is covered in the subsequent section.

IV. Thief: The Dark Project

A classical toy example when considering epistemic planning, is the video game *Thief: The Dark Project™* by Eidos Interactive (1998). The player, being the thief, avoids being detected by guards, exploiting their possibly mistaken beliefs about the thief's presence. This game can be extended to epistemic logic, by having the guards include beliefs about what the thief believes, and beliefs about what the thief believes he believes.

A. Scenario

To illustrate this extension we consider a minimalistic example scenario with a thief and a guard, and a limited number of possible events. The possible events are the following:

- n_t/n_g : The thief/guard makes some noise.
- s_t/s_g : The thief/guard is seen from behind.
- f : The thief and guard face each other.

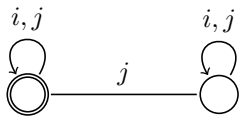
of which the epistemic effects are:

- n_t : The guard learns that the thief is present. The thief learns that if a guard is present, the guard learns that the thief is present.
- s_t : The guard learns that the thief is present, while the thief believes nothing has happened.
- f : The thief and the guard commonly learn that both are present.

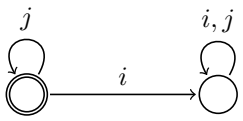
The effects of n_g and s_g are analogous.

As a simplification, all events to occur with a probability, such that we do not consider the actual whereabouts of the characters involved. We also assume that the thief and the guard become aware of the each others presence if they hear a noise they did not cause themselves. Furthermore, if the thief makes a noise, the guard will indeed hear it, and the thief will know that the guard knows.

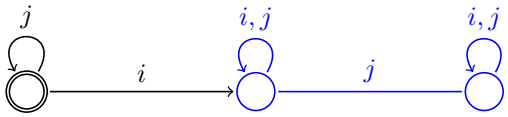
This scenario can be formalized in DEL, using the atomic propositions p_t and p_g, denoting that the thief and the guard is present respectively. Let $\mathcal{I}$ be the pointed model of the initial



(a) Box model: Epistemic model (M, s_1) with two possible possible lock combinations where, $M = (\{ c_1, c_2 \}, \pi, \mathcal{K}_i, \mathcal{K}_j)$.



(b) Secret message: Event model $(\mathcal{E}, e_1)$ for secret message in the box model. Here $c_1 \wedge \neg c_2$ is the precondition for e_1.



(c) Updated model after secret message, $((M \otimes \mathcal{E}), (s_1, e_1))$. Blue indicates the part isomorphic to figure 1a.

Fig. 1: Locked box example for dynamic epistemic model

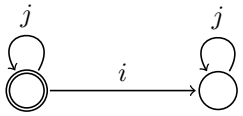


Fig. 2: Private message event model for j in locked box model. Note similarity with figure 1b.

Fig. 3: Public announcement in locked box model

situation, and the set $E = \{\mathcal{N}_t, \mathcal{N}_g, \mathcal{S}_t, \mathcal{S}_g, \mathcal{F}\}$ be pointed event models of the events described above. A graphical representation of these pointed event models is depicted in figure 4.

B. Epistemic Planning

Artificial intelligence is used in games to create more life-like non-player characters (NPCs), for instance by introducing goals and ways of realizing them. One of those AI engines is the *Radiant AI* used in the role-playing game *The Elder Scrolls: Oblivion™* (Bethesda Softworks 2006), where each individual NPC has a goal or purpose to fulfill. This approach is however limited in terms of epistemic reasoning, which was quite evident during the testing phase of the game AI:

> One [non-player] character was given [by the testers] a rake and the goal "rake leaves"; another was given a broom and the goal "sweep paths", and this worked smoothly. Then they swapped the items, so that the raker was given a broom and the sweeper was given the rake. In the end, one of them killed the other so he could get the proper item. [12]

By introducing a notion of epistemic knowledge, and being able to act upon that knowledge, these two NPCs could simply swap items instead of resolving to violence.

In our scenario on the other hand, violence is the only option, and the thief and the guard have three possible attacks to perform depending on the situation:

- **ambush**, with precondition $p_t \wedge \neg K_t p_g$.
- **trick**, with precondition $p_t \wedge K_t p_g \wedge \neg K_t K_g p_t$.
- **rush**, with precondition $p_t \wedge K_t K_g p_t$.

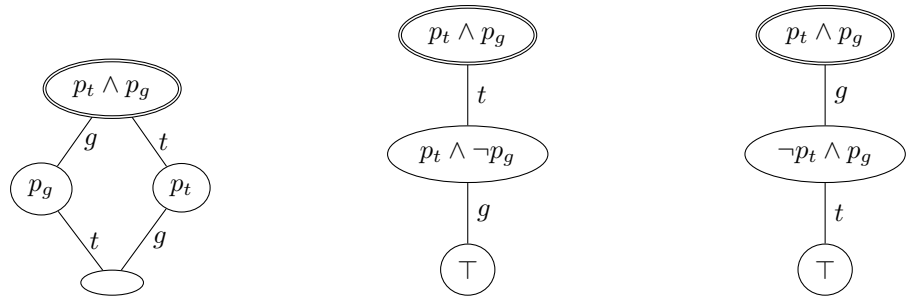
for the guard, and analogously for the thief. Note that the preconditions for **rush** can be simplified to common knowledge about both agents being present ($Cp_t \wedge Cp_g$).

The guard prefers to ambush the thief, however if the guard suspects that the thief has noticed him (the guard), he can try to trick the thief by playing stupid. Doing so will only work if the thief does not know that the guard has noticed him (the thief), having a face-to-face encounter as last resort. To successfully ambush the thief, the guard has to avoid making any noise or being seen while sneaking up on the thief. As the events are out of the agents' control, they have to re-plan accordingly when an event occurs and their knowledge is updated.

An example run of the scenario goes as follows:

1) The thief is seen by the guard (s_t). The guard plans on ambushing the thief.
2) The guard suddenly makes a noise (n_g). Now the thief knows the guard is present. Since the thief does not know that the guard has noticed him, he plans on ambushing the guard. Meanwhile, the guard knows he has been noticed, thus re-plans by trying to trick the thief instead.
3) The thief makes a noise as well (n_t). At this point they both know they are aware of each others presence, resulting in face-to-face combat by rushing towards one another.

A more complex and realistic example would include the triggering of events, such that the guard can intentionally make a noise (n_g), let himself be seen from behind (s_g) and step out to provoke a face-to-face encounter (f). As a result, there would be multiple applicable plans for the guard after noticing the thief. For instance, he could potentially plan to make a



(a) $\mathcal{I}$: initial situation (b) $\mathcal{N}_t$: thief makes noise (c) $\mathcal{N}_g$: guard makes noise

(d) $\mathcal{S}_t$: thief seen from behind (e) $\mathcal{S}_g$: guard seen from behind (f) $\mathcal{F}$: face-to-face encounter

Fig. 4: Pointed event models for the initial situation and the possible events. Bidirectional accessibilities are represented by undirected edges, and reflexive accessibilities are implicit in models without directed edges.

noise and try to trick the thief ($P_1 = n_g$, trick) or simply ambush the thief ($P_2 =$ ambush). In both cases, there are several events the guard should avoid in order to successfully execute his plan. In case of P_1, the guard must avoid them facing each other, and more delicately, the guard must avoid having the thief make a noise (n_t). If the thief makes a noise, the thief will immediately learn that the guard is aware of his presence, and the guard will not be able to trick the thief. In the simpler case P_2, the guard must avoid even more events, namely being seen from behind (s_g), making a noise (n_g) and having them face each other (f). By considering all plans ending with one of the attack actions up to a certain length, the space of potential plans is still of manageable size, however for more advanced examples, the planning space can quickly become unfeasible.

C. Implementation

The simple *Thief* example is implemented in AgentSpeak, a logic based agent-oriented programming language used to model agents' beliefs, desires and intentions. Its grammar is similar to Prolog, having capitalized words and letters as variables, and non-capitalized as literals with arguments or simply atoms. Each agent has a belief base, comprising atoms and literals, which is modified through occurring events and by executing plans. Plans and events are defined by a name, [(arguments)], [: preconditions] and [<- postconditions], using the following syntax (elements enclosed with [] are optional):

```
+!name(argument, ..., argument) :
precondition & ... & precondition <-
postcondition; ...; postcondition.
```

The difference between plans and events are the inclusion of `!` in front of the plan name as seen in the code above, since plans are executable by agents as `!name(argument, ..., argument)`. The actions a guard can perform to attack the thief can then be modelled using the following plans:

```
+!action(ambush) : p(t) &
    not know(t, p(g)).
+!action(trick ) : p(t) & know(t, p(g)) &
    not know(t, know(g, p(t))).
+!action(rush  ) : p(t) &
    know(t, know(g, p(t))).
+!action(none  ).
```

such that `!action(A)` will try executing all AgentSpeak plans matching the name and the argument count top down until one of the plans succeed. If the guard's belief base comprise `p(t)` and `know(t, know(g, p(t))`, the variable `A` will be unified with `rush`, being the first applicable plan from the top. Note that none of these actions are defined with any post-conditions, since we are only interested in the planning and reasoning with regards to high-order beliefs.

The guard is unable to observe the event in which the thief sees him from behind (`seen(g)`), thus all events he consider possible are modelled as follows:

```
+noise(t) <- +p(t);
    +know(t, know(g, p(t))).
+noise(g) <- +know(t, p(g)).
+seen (t) <- +p(t).
+face     <- +p(t); +know(t, p(g));
    +know(t, know(g, p(t))).
```

such that `+belief(B)` will add the literal to the agent's belief base. As seen above, the `face` event acquires the most

knowledge, having the guard learn that the thief is present; that the thief knows the guard is present; and that the thief knows that the guard knows he is present. Note that the addition of beliefs correspond to the epistemic effects of the events (from the guard's perspective) given in the scenario description.

Having a third almighty entity, *nature*, randomly making events occur each step, the guard and the thief will simultaneously plan their actions according to their current knowledge. As no actions have post-conditions, they will continue re-planning until they commonly learn that both are present, resulting in the inevitable face-to-face encounter were both agents rush towards one another. The example run of the scenario described in the previous section can be produced with the following output, given that the same events occur in identical order.

```
[nature] seen(t)
[guard] ambush
[thief] none

[nature] noise(g)
[guard] trick
[thief] ambush

[nature] noise(t)
[guard] rush
[thief] rush
```

While this example shows the results of epistemic reasoning in games, the next step would be to extend the implementation to include triggerable events in agents' action repertoires, allowing for more sophisticated plans. By doing so, the characters would be able to act more life-like, facilitating the surrounding environment to achieve their goals (future work).

This extension does however require a notion of executable events, which AgentSpeak does not provide. To do so, the articles [12][8] propose the use of a knowledge module, managing epistemic effects similar to how physics engines manage the laws of physics in games. This module is effectively a DEL model. Doing so, epistemic formulas can be evaluated and their effects interpreted, giving NPC scripters a *black box* to directly use notions of belief and knowledge in their programs. However, for the implementation to remain tractable, a certain subclass of formulas needs to be identified as relevant, thus implementing a full human-like knowledge module is out of the question.

D. DEL Planning Limitations

Having established itself as a standard conceptual model for epistemic situations and change, DEL makes a promising framework representing the knowledge- and belief-based parts of computer games. DEL planning problems can easily be defined for single-agent scenarios, assuming no interfering actions of other agents, and adversarial scenarios assuming complete opposition. More interestingly however, is a general strategic DEL planning problem, where agents take actions of other agents into account, and may to some extent try to anticipate them. Existing work attempts to do so by modelling goals and preferences [11], although these scenarios are far less clear, and requires more thorough research.

The main technical issue that arises, is that in general it may be unfeasible to solve the DEL planning problem altogether. The search space can easily be infinite, or even too large if it is not. In multi-agent epistemic planning problems, there is in general no upper bound on the size of the reachable epistemic states, and is undecidable even without common knowledge [3]. One of the most studied problems in epistemic planning based on DEL is the complexity of the plan existence problems. The plan existence problem is a decision problem regarding a class of planning tasks X, and whether there exists a solution to a planning task Π for $\Pi \in X$. The problem is already undecidable with two agents, no common knowledge, and even without post-conditions [2].

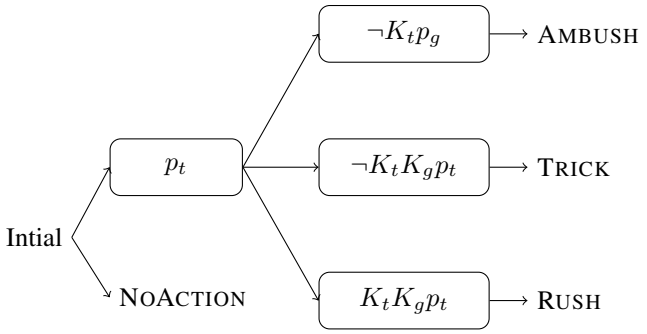


Fig. 5: Decision tree for the guard based on his believes about the world. Each layer in the tree should be read top to bottom, and the first applicable action should be chosen. The decision tree for the thief can be created analogously.

V. GAME THEORY

The progress of this game is based on the different events described happening randomly. Each of the agents in the scenario does not have any control over the events that they make a noise, see the other agent, or see each other, and are therefore modelled as random events that will occur with given probabilities. The agents themselves can therefore only try to respond with appropriate actions to their own favor when they acquire knowledge about the world.

As such, which decisions the guard and thief should take in this game depend only on which world they believe they are in. As mentioned earlier, the general planning problem quickly becomes undecidable. Hence to create a valid plan, the agents should use a greedy strategy that will give them the best outcome. In figure 5, the decision tree for the guard is illustrated (the thief's being similar). This decision tree describe each of the actions the guard is capable of, and the priority of their execution from top to bottom, based the guard's knowledge of the world. The guard will start in a initial state. If he does not know the thief is present, the guard will do nothing. However, if the guard does, which action he should perform depends on what he knows about the thief's knowledge.

If the guard knows that the thief does not know the guard is present ($K_g \neg K_t p_g$), the guard is allowed to ambush the thief. If the guard instead knows that the thief does not know that the guard knows the thief is present ($K_g \neg K_t K_g p_t$), the guard is able to trick the thief. Lastly, if the guard and thief commonly knows the other is present, they have no other option than to rush each other.

This model does create one problem: if both agents have seen each other from behind (both s_t and s_g have occurred), they mutually know the other agent is present, but also that the

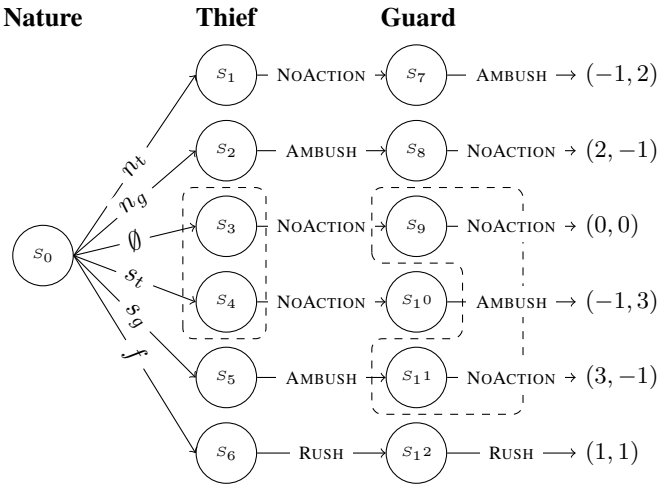


Fig. 6: Illustration of a step in the game, where both players have no prior knowledge. The order of the thief and guard are indifferent.

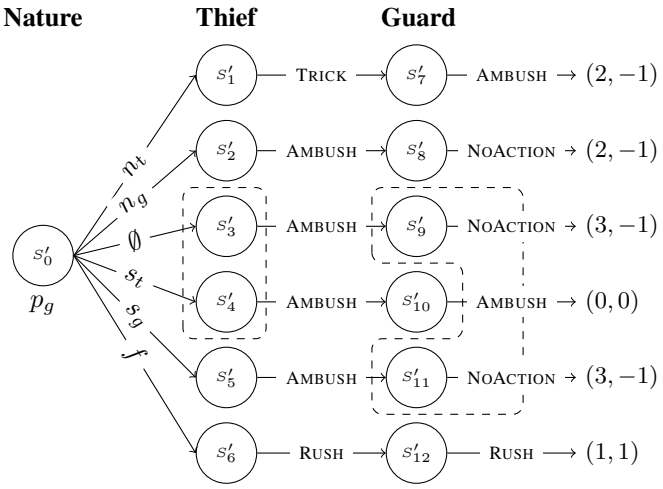


Fig. 7: Illustration of a step in the game, where everybody already knows that the guard is present. The order of the thief and guard are indifferent.

other agent is not aware of their own presence. This is false knowledge, which should not be possible. It would result in both of the agents trying to AMBUSH the other, which is a kind of paradox and should be impossible. Because of this, all of the agents' knowledge should instead be modelled as *beliefs*, meaning knowledge that can be false. However, for simplicity this will be disregarded.

A. Planning without prior knowledge

One step in the game can be modelled by letting *nature* take actions before both of the players turn. The actions taken by nature will correspond to the events that can happen in the game: either of the players makes noise, see the other, or they see each other. To simplify the game, it is assumed only one of these events can happen in each step, each of which will happen with some probability. In figure 6 one step of the game is modelled, where it is assumed that none of the players have any prior knowledge. Because of this assumption, the tree does not catch the trick outcome, as this would require one of the players to know that the other one is present. Note that in figure 6 every node has an implicit NOACTION, but both to keep the figure simple and because the NOACTION-strategy is weakly dominated by the strategy presented in the figure, these have been left out.

In the game, the players can take actions simultaneously. However, the situation is transformed into a sequential model, with either of the players doing their action first, without reflecting on the action taken by the other player. Hence the tree in figure 6 could just as likely have the order of the thief and guard switched.

Based on the different actions each of the agents can perform, the outcome is scored accordingly. The score is depended on which attack is selected to perform. Ambushing the other agent will give the highest probability of a kill, especially if the agent being ambushed have no idea that the other agent is present. Rushing the other agent gives a worse chance of winning, but is still better than doing nothing. Being attacked is always scored the same negative value.

B. Planning with prior knowledge

In the previous model it was assumed that the agents had no prior knowledge. To describe more advanced situations, the game tree from before have to be extended to handle knowledge from previous steps. There are three situations to analyze: no knowledge from previous step, both agents have learned that the other is present, and one agent has learned that the other is present, while the other has not. Of course more complex knowledge could transfer between steps, such as knowledge about the other agents knowledge, but the outcomes of these transfers can also be found in the following model.

The first two cases are trivial. If there is no knowledge from the previous step, the situation is the same as before, and the model from before describes this situation. Secondly, if both have learned the presence of the other, there is no more information for them to learn.

The interesting case is when one of the agents have learned that the other agent is present, but the other has not learned anything, as illustrated in figure 7. Many of the situations remain the same, however, looking at the case were n_t happens, the thief is now able to TRICK the guard because of its knowledge of the guards presence. The second interesting case is when no event happens, as the thief will still be able to perform an AMBUSH action. However, as mentioned earlier, this creates the situation where both agents will try to AMBUSH the other. However, despite this figure 7 also shows that the thief is in a quite superior position, after learning the presence of the guard.

C. Information categories

For both of the agents in the scenario, some states are indifferent. Both of the agents are unable to distinguish between the situation where nothing has happened and where the other player has seen them from behind. In figure 6, the thief sees state 3 and 4 as the same, while the guard is indifferent between state 9 and 11. Hence the game is of imperfect information. In the second model, figure 7, the thief

does have the same information in state 5 as in state 4 and 3, making these states essentially the same from the thief's point of view. However, because the thief have seen the guard a second time, the thief should be able to distinguish these events, as it have had its knowledge confirmed. It also informs the thief that the guard did not see the thief from behind, which is useful information to the thief.

Agents also have asymmetric information in the game, primarily because they do not know if the other is present, but also because agents are able to see the other without the other agent knowing. As nature moves first, and both players are not always able to see the action it takes, the game is also of incomplete information.

In the single step version of the game, the agents have certain information, as nature does not take any actions after the agents, thus the outcome is deterministic. However, this only models a single step in the game, which might not be enough to end the game. This single step model could therefore be repeated until a resolution of the game is found, with knowledge carrying over between each step, making the information uncertain.

VI. Conclusion

Having introduced notation and concepts for dynamic epistemic logic, we were able to apply these concepts to a simple game and analyze it in terms of possible outcomes. DEL allows us to model epistemic situations and change, defining event models and product update to represent the effects of epistemic events. The game comprise several of these events, each resulting in different knowledge for the agents involved, being able to utilize the acquired knowledge to defeat their opponent. By analyzing the game, we were able to identify the superior strategy in the simplified scenario, where we only plan one step ahead. A natural extension would be to implement and analyze the extended scenario, where events are included in the agents' action repertoires. As a result, the agents would have improved planning capabilities, allowing for more interesting game strategies and outcomes. Furthermore, by introducing specific probabilities to different events, say the thief is not very discrete or the guard always makes noise, the strategy would have to be changed accordingly.

A generalized DEL framework would simplify this extension, by using a knowledge module keeping track of event models and updating them accordingly when events occur. However, most interesting applications are not about knowledge, but rather about belief, and further work would include developing a framework for dynamic doxastic logic as initiated by Kennerly et al. [7].

References

[1] Mordechai Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[2] Thomas Bolander. "A Gentle Introduction to Epistemic Planning: The DEL Approach". In: *arXiv preprint arXiv:1703.02192* (2017).

[3] Thomas Bolander and Mikkel Birkegaard Andersen. "Epistemic planning for single- and multi-agent systems". In: *Journal of Applied Non-Classical Logics* (2011).

[4] Jan van Eijck. "a demo of epistemic modelling". In: *Interactive Logic* (2007), p. 303.

[5] Jan van Eijck. *Analyzing Communication with Dynamic Epistemic Logic*. URL: https://pdfs.semanticscholar.org/4b4e/c6eab9c41e6cd51233ef06f337f2135e4e83.pdf (visited on 12/01/2017).

[6] Ronald Fagin et al. *Reasoning about knowledge*. MIT press, 2004.

[7] Ethan Kennerly, Andreas Witzel, and Jonathan A Zvesper. "Thief belief". In: *Games Innovations Conference, 2009. ICE-GIC 2009. International IEEE Consumer Electronics Society's*. IEEE. 2009, pp. 169–172.

[8] Benedikt Löwe, Eric Pacuit, and Andreas Witzel. *Planning based on dynamic epistemic logic*. Tech. rep. Technical Report PP-2010-14, Institute for logic, Language and Computation, Universiteit van Amsterdam, 2010.

[9] Eric Pacuit. "Dynamic Epistemic Logic I: Modeling Knowledge and Belief". In: *Philosophy Compass* 8.9 (2013), pp. 798–814. ISSN: 1747-9991. DOI: 10.1111/phc3.12059. URL: http://dx.doi.org/10.1111/phc3.12059.

[10] Floor Sietsma and Jan Van Eijck. "Message passing in a dynamic epistemic logic setting". In: *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge*. ACM. 2011, pp. 212–220.

[11] Johan Van Benthem and Fenrong Liu. "Dynamic logic of preference upgrade". In: *Journal of Applied Non-Classical Logics* 17.2 (2007), pp. 157–182.

[12] Andreas Witzel, Jonathan A Zvesper, Ethan Kennerly, et al. "Explicit Knowledge Programming for Computer Games." In: *AIIDE*. 2008.