Thief Belief
– Extended Abstract –

Ethan Kennerly
Interactive Media
School of Cinematic Arts
University of Southern California
University Park, LUC-310B
Los Angeles, CA 90089-2211

Andreas Witzel
ILLC, University of Amsterdam
Science Park 904
1098XH Amsterdam, Netherlands
and CWI, Science Park 123,
1098XG Amsterdam, Netherlands

Jonathan A. Zvesper
ILLC, University of Amsterdam
Science Park 904
1098XH Amsterdam, Netherlands
and CWI, Science Park 123,
1098XG Amsterdam, Netherlands

Overview

The video game Thief: The Dark Project™ (Eidos Interactive 1998) is themed as a game of stealth, in which the player (the thief) avoids being detected by computer-simulated guards. Essentially, the player exploits the, possibly false, beliefs of the guard regarding the thief's presence. Due to the simplicity of the guard's control program, the guard's beliefs are in practice perceived as either believing that the thief is, or may be, near (having seen him or become suspicious in some other way) and acting accordingly, or not.

We conjecture that the entertainment value of a typical Thief scenario would be enhanced by a guard that acts not only depending on his own beliefs about the facts in the world, but also depending on what he believes the thief believes, including what he believes the thief believes he believes.

Besides making the interaction more realistic, we believe that dealing with, and possibly exploiting, higher-order beliefs is intrinsically enjoyable. This is not news to the logic community, in fact there exist detailed formalizations of the epistemic mechanisms involved in real-world games of knowledge and suspicion such as Cluedo (Ditmarsch 2000). However, so far there has not been much focus on the complementary direction of putting such formalizations to work to support a computer simulation of a virtual game world.

In this paper, we will substantiate our previous discussion (Witzel, Zvesper, and Kennerly 2008) by describing an actual implementation of the pseudo-code given there and providing a detailed description of an experiment setup intended to test our conjectures about the increase in entertainment value resulting from exploiting higher-order beliefs. Before conducting the actual experiment, we are planning a small pilot study with 6 subjects in the control group and 6 subjects in the experimental group.

Scenario

In the control scenario, when a guard has noticed a thief, the guard attacks or calls an alarm as per its scripted behavior. In the experimental scenario, the guard models the beliefs of the thief based on his own current beliefs and past observations, and acts accordingly.

In the original game, a common tactic is to shoot an arrow against a wall, in order to cause a noise which will attract the guard (who has a behavior rule along the lines of "if I hear a noise, I go there and look around"). It is questionable how innocent such a noise is, and an actually reasoning guard would probably rather conclude that it is high time to ring the alarm instead of sniffing around in the bushes. While this was likely a conscious design decision to give the player an easy and obvious way to outwit that guard, we think that a belief-based approach can provide a more flexible solution.

Instead of rigidly connecting events to resulting behaviors, we therefore introduce beliefs as an abstraction layer. The behaviors are defined depending on the beliefs, and these beliefs are modeled independently.

For example, if the guard believes that the thief is present, but the guard believes that the thief does not believe that the guard is present, the guard tries to ambush the thief, rather than attacking him openly or ringing the alarm. This and a few more behavior rules constitute the guard control program in our experimental scenario, shown in Listing 1. We will specify the events and the semantics of beliefs later on.

This approach removes from the scripter the burden of having to decide exactly which events cause what, and facilitates adjustments and more complex dependencies. For example, in our scenario a more clearly innocent noise such as breaking a twig will induce the guard to believe that there is a thief (who accidentally caused that noise), which will then trigger the behavior rule that says to ambush him.

Experiment design

We stipulate some necessary conditions for entertaining deception:

1. The exploitation of the higher-order belief is intrinsically enjoyable.
2. The player knows there is a reward for modeling the other agents' (higher-order) beliefs.
3. The player expects the other agents to model the player's beliefs.

Based on these assumptions, we will run a small population pilot study among volunteers who are randomly assigned to a control group or experimental group.

Before play, the player is surveyed for the last video game that they played and their level of enjoyment on a five-point Likert scale, and their interest in playing that game again.

```
if Believes(guard, thief_present):
    if Believes(guard, not Believes(thief, guard_present)):
        guard.ambush(thief)
    elif Believes(guard, not Believes(thief, Believes(guard, thief_present))):
        guard.attack_or_alarm_inconspicuously()
    else:
        if not alarm_active:
            guard.alarm_quickly()
        else:
            guard.attack(thief)
```

Listing 1: Guard control program in Python style

Text directly informs the player that the agents have a mental model. Without explanation, a player may fail to model the mechanisms underlying the behavior of agents in a video game. In the control scenario, the player is informed of the guard and that it will respond to noticing the thief. In the experimental scenario, the player is informed that the guard will ambush, attack, or call for help depending on how the thief behaves.

The session of play occurs on a personal computer using an animated two-dimensional prototype of the scenario. The graphical depiction and other media are identical. The setting and characters are identical. The player is also informed about the input and output conventions, which are identical for both the control group and experimental group. The difference is isolated to the behavior of the artificial agent.

During play, the player's facial expressions, vocal responses, and body language are observed. Signs of (plausibly) spontaneous emotions, such as chuckles, grins, furrowed brows, crossed legs are noted.

We conjecture that our modest belief engine encourages the original game's tactics of misleading a guard. It also encourages pretending in another way: The thief may play stupid and let himself be seen from behind by the guard, who then again thinks he can do something sneakier than ringing the alarm.

We also conjecture that our scenario encourages trepidation. A guard who has actually noticed the thief without being noticed himself will act inconspicuously while preparing a counter-attack. The blackjack is the weapon for subduing, which has a short reach. A guard that pretends to not have noticed, will swing around and stab with his sword first before the player's blackjack is in range.

Overall, we conjecture that the average enjoyment and interest in the experimental group exceed those of the control group.

To test our conjectures, after play the player is surveyed for their level of enjoyment on a five-point Likert scale, and their interest in progressing in this game. The player is also surveyed on open-ended questions about the tactics they employed, and what stood out in their experience.

Knowledge module

There are various ways in which relevant beliefs can come about: by causing or hearing noise, seeing the other one from behind, or facing each other. When scripting the behaviors, the programmer need not worry about how exactly the beliefs came about, he simply uses the familiar concept of belief in order to express the rules on a high level.

It is the job of the *knowledge module* to take care of maintaining and updating the agents' knowledge,[1] taking into account whichever events occur in the virtual world. As an agent's control script is executed, the knowledge module is queried and determines the truth value of any given formula.

We assume that the scene starts with thief and guard present and no events having occurred, and that agents cannot enter or leave. Put differently, the guard will not leave the scenario, and if the player leaves, the scenario ends. Note that this assumption is not inherent to our approach and serves mostly to avoid unnecessary complications. In the context of a computer game it is not too unnatural: Game worlds often are simulated per scene, and a scene starts and ends whenever the player enters or leaves.

In our experiment, we consider the following kinds of events:

- b_t, b_g: The thief, respectively the guard, sees the other one from behind.
- n_t, n_g: The thief, respectively the guard, makes some noise. We limit this to "relevant" events in the sense that, e.g., n_t can only occur if the thief is present.
- f: Thief and guard see each other face to face.

The epistemic effects of these events are as follows:

- b_t: The thief believes that the guard is present; the guard never assumes this event to occur, so this event does not change the guard's beliefs.
- n_t: The guard believes a thief is present; the thief believes that, if a guard is present, that guard believes that the thief is present.
- f: Thief and guard commonly believe that both are present.

In addition, these effects are commonly believed. For now, we assume that events are not forgotten. Note that a consequence of all this is that after both n_g and n_t have occurred, guard and thief commonly believe that both are present.

[1] In our case beliefs rather than knowledge, but we will stick with the name 'knowledge module'.

To model this situation formally, we use a modal logic model with history-based semantics (Fagin et al. 1995). We introduce two **propositions**, p_t and p_g, with the reading that the thief, respectively the guard, is present. A **valuation** is a function that assigns either true or false to each of these propositions, and can be denoted as a set V containing those propositions that are to be assigned true.

The set of **events**, as above, is $E = \{b_t, b_g, n_t, n_g, f\}$. A **history** H is a sequence of events. For a history H, by $Ag(H)$ we denote the set of agents involved in the events in H, i.e. $Ag(H) \subseteq \{t, g\}$, where naturally both t and g are involved in any of $\{b_t, b_g, f\}$, and only t (resp. g) is involved in n_t (resp. n_g). We also write $H - e$ for any $e \in E$ to denote the history obtained by removing all occurrences of e from H. The empty history is denoted by ϵ.

A pair (V, H) is a **state** (or **possible world**) if and only if $Ag(H) \subseteq V$. So the described initial situation is represented by the state $(\{p_t, p_g\}, \epsilon)$.

We now define accessibility relations $\dashrightarrow_t$ and $\dashrightarrow_g$ between states in our history-based model. Intuitively, $(V, H) \dashrightarrow_t (V', H')$ means that in state (V, H), t considers it possible that the state is (V', H'). According to the intuitions described above, we define these relations as follows:

$$(V, H) \dashrightarrow_t (V', H') \text{ iff } \begin{cases} t \notin V' \text{ and } H' - n_g = \epsilon & \text{if } t \notin V \\ t \in V' \text{ and } H' = H - b_g & \text{if } t \in V. \end{cases}$$

Note that in the first of these two cases, n_g is the only event which can occur in H', so the condition boils down to saying that H' may be any sequence of n_g. The second case captures the intuition that t does not assume the possibility that he is being seen from behind. The relation $\dashrightarrow_g$ is defined analogously.

We consider the doxastic language consisting of formulas of the following form:

$$\varphi ::= p_t \mid p_g \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid B_t\varphi \mid B_g\varphi \ .$$

The semantics is standard relational modal semantics, see (Blackburn, de Rijke, and Venema 2001).

If we merge all doxastically equivalent states, that is, all states that make the same formulas true, our model can be depicted as in Figure 1. [2]

Note that, in contrast to our starting point from history-based semantics, this representation is static in the sense that it does not grow (or shrink) over time, as the world evolves and events occur. It is straightforwardly represented in any programming language (our implementation uses Python), and the knowledge module simply needs to keep track of which one is the current state.

Since in a computer game there is a central place where the game world is simulated, only one such model needs to be maintained, even in a scenario with more agents. Rather than maintaining a separate model inside each agent, this central model is fed with the events occurring in the game world, and queried by the agents' programs. This also means that the evaluation of formulas can be done efficiently since it corresponds to model checking, rather than testing validity as in a truly distributed setting.

Finally, note that while agents can have false beliefs, there is no need for elaborate belief revision mechanisms in our simple scenario, since there is no way for the agents to notice their false beliefs.

[2] This is the so-called 'bisimulation contraction' of our original model. Note that in our case, if (V, H) and (V', H') are doxastically equivalent, so are (V, He) and $(V', H'e)$ for any event e.

Technical discussion

The knowledge module we presented is a centralized implementation of the approach described in (Witzel and Zvesper 2008). The fact that it is centralized together with the simplicity of our scenario remove the necessity of finding restrictions of the doxastic language or other optimizations: About any conceivable model and implementation would be trivially tractable.

However, our scenario still shows the advantages of the modular approach which consists in introducing beliefs as an abstraction layer and separating the belief model from the agent's control program. The knowledge module can be refined independently to cope with new events or extensions, such as probabilistic beliefs, with no need to change the behavior scripts. While one could implement some ad-hoc tracking of events in the agent control program, this is error-prone and difficult to maintain, as already in our small test scenario the exact dependencies on events are getting confusing: What does "if n_t or b_g have occurred, but neither n_g nor f have, then..." mean?

Of course a knowledge module may also need debugging, but this can be done separately from the rest of the program. To this end, it would be interesting to define a $Why()$ function, which should give a (useful) explanation of why a given belief formula holds in a given state.

One extension that is easy to incorporate into our knowledge module are decaying events, or unstable states: If the guard makes some noise, like whistling (which he obviously would only do if he does not believe a thief is present), he might have forgotten about that when he 10 minutes later sees the thief from behind, and therefore *not* conclude that the thief believes the guard is present. This could be modeled by a spontaneous transition from state 5 to 2 after some time.

The advantages of our finite-state-machine-like representation is that it is straightforwardly represented using a very simple data structure, and that it is a static, pre-computed model, so there is no expense at run-time and no surprises from uncontrolled growth. However, for bigger scenarios, it may not be feasible to keep the whole model in memory all the time. Mechanisms to generate the model only locally as needed, and strategies for expanding, shrinking, or swapping out unused parts of the model may be necessary.

A very promising alternative is Dynamic Epistemic Logic (DEL). Instead of using one model to represent all possible ways in which the world may evolve, one starts with a model representing only the initial situation. Events are represented as so-called 'update models', which are applied to the current model as the events occur, changing the model to reflect the resulting situation.

It is then desirable to ensure that the model will not grow indefinitely. In our scenario, this can be achieved by show-

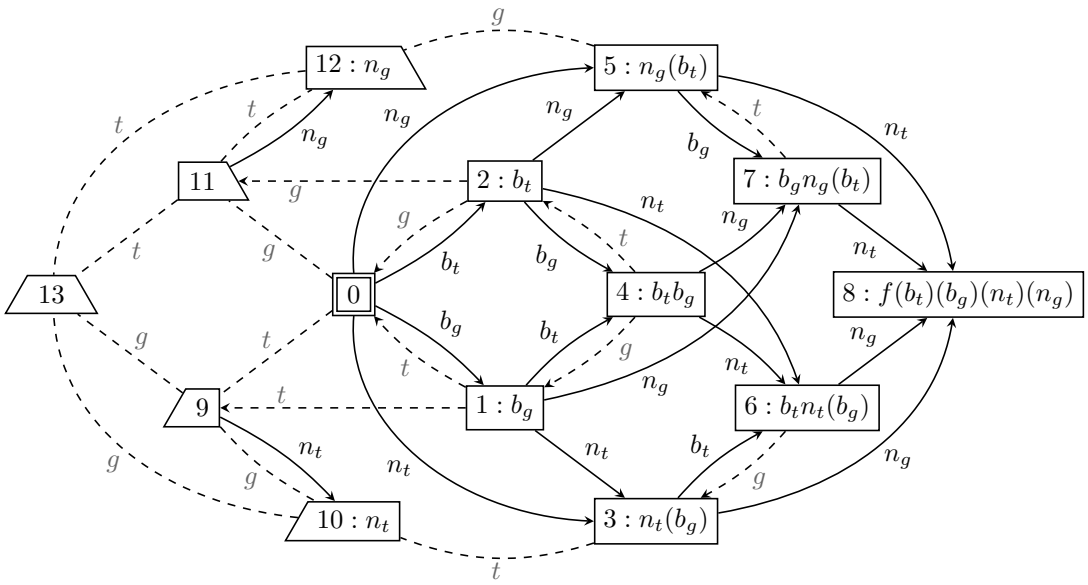

Figure 1: Representation of our doxastic model. The numbered nodes represent states and are annotated with one of the equivalent possibilities of what events have (optionally) taken place. Boxed states have $V = \{p_t, p_g\}$, a missing left corner means $p_g \notin V$, and a missing right corner means $p_t \notin V$. State 0 represents the initial situation. Solid arrows with black labels show event transitions, dashed arrows and edges (the latter corresponding to bidirectional arrows) with gray labels show accessibility relations. The following are omitted for clarity: reflexive transitions for each optional event at each state; f-transitions from each boxed state to 8; and reflexive accessibilities, except for g in states 2 and 6 and for t in states 1 and 7.

ing that the events are commutative and idempotent in an adequate sense, and that the update mechanism preserves minimality with respect to bisimulation. It would be interesting to look at such properties of DEL models and events in a more general way.

Besides this 'online' use, DEL could also be used simply as a description language for initial situation and events, from which a static model similar to ours could be precomputed 'offline'. Again, considerations about the size of the resulting model, or language restrictions to ensure certain complexity bounds, would be relevant.

An interesting suggestion that would also simplify reasoning is to consider so-called 'interaction axioms' between the belief modalities of the players. For example, in (Lomuscio and Ryan 1998) the authors show that (2WD) is valid on two-player 'hypercubes', i.e., models that are based on the Cartesian product of an interpreted system's state space:

$$\Diamond_1 \Box_2 p \implies \Box_2 \Diamond_1 p. \qquad \text{(2WD)}$$

In a different context (Ditmarsch and Labuschagne 2007) consider interaction axioms that characterise different Theories of Mind, including 'autistic' and 'deranged'. It is an interesting question whether interaction axioms that capture agents that are *fun* to interact with might exist.

Another point worth noting is that we do not distinguish between *knowledge* and *true belief*. There are many philosophical discussions concerning differences between these two notions, and formally they are generally taken to be different as well. Beliefs should really be *revisable* for example, which is something we have not considered here. A possible future direction of research would be to use models and languages that take both belief and knowledge into account, for example along the lines of (Shoham and Leyton-Brown 2009, Chapter 13). Some actions would then generate knowledge, while some would (only) generate belief. This corresponds to the distinction van Benthem (2007) draws between 'hard' and 'soft' information. For example, if you *see* something then you might be said to know it, but if somebody you don't entirely trust tells you something then you might only believe it.

Finally, it may be desirable to get rid of manually designed behavior rules altogether, and let the agents act purely based on their beliefs and some abstractly defined goals. In order to accomplish this, a very promising approach is to combine a game world that uses deductive planning, e.g. (Magnusson and Doherty 2008), with a belief model such as the one we discussed. In this way, the planning agents would be enabled to reason about the epistemic preconditions and consequences of their actions and the epistemic parts of their goals.

A very long-term vision is then to have an artificial guard that by himself adopts behaviors similar to the ones we described, or to the tactics we conjectured for the human thief.

Acknowledgements

We thank Martin Magnusson, Cédric Dégremont and three anonymous referees for discussion and helpful suggestions. The second and third authors were supported by a GLoRiClass fellowship funded by the European Commission (Early Stage Research Training Mono-Host Fellowship

MEST-CT-2005-020841).

References

[Benthem 2007] Benthem, J. v. 2007. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics* 17(2):129–155.

[Blackburn, de Rijke, and Venema 2001] Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*. Cambridge University Press.

[Ditmarsch and Labuschagne 2007] Ditmarsch, H. P. v., and Labuschagne, W. A. 2007. My beliefs about your beliefs: a case study in theory of mind and epistemic logic. *Synthese* 155(2):191–209.

[Ditmarsch 2000] Ditmarsch, H. P. v. 2000. *Knowledge games*. Ph.D. Dissertation, Groningen University. ILLC Dissertation Series 2000-06.

[Eidos Interactive 1998] Eidos Interactive. 1998. Thief: The Dark Project. `http://www.eidosinteractive.com/games/info.html?gmid=34`.

[Fagin et al. 1995] Fagin, R.; Halpern, J. Y.; Vardi, M. Y.; and Moses, Y. 1995. *Reasoning about knowledge*. MIT Press.

[Lomuscio and Ryan 1998] Lomuscio, A., and Ryan, M. 1998. Ideal agents sharing (some!) knowledge. In *Proceedings of Proceedings of the 13th European Conference on Artificial Intelligence*, 557–561. John Wiley & sons.

[Magnusson and Doherty 2008] Magnusson, M., and Doherty, P. 2008. Logical agents for language and action. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-08)*.

[Shoham and Leyton-Brown 2009] Shoham, Y., and Leyton-Brown, K. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

[Witzel and Zvesper 2008] Witzel, A., and Zvesper, J. A. 2008. Epistemic logic and explicit knowledge in distributed programming (short paper). In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*.

[Witzel, Zvesper, and Kennerly 2008] Witzel, A.; Zvesper, J. A.; and Kennerly, E. 2008. Explicit knowledge programming for computer games. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-08)*.