Knowledge and games : theory and implementation
Witzel, S.A.

Citation for published version (APA):
Chapter 7

Outlook

One main topic of this dissertation has been what we called explicit knowledge programming: giving artificial agents access to the knowledge one can theoretically ascribe to them, by way of providing epistemic statements in a programming language, made feasible by considering scenarios with well-defined restrictions. We believe that this approach is worth exploring in a more systematic way.

As illustrated in Chapter 4, simulations of virtual worlds, and in particular computer games, form a rich playground and testbed for formalisms and implementations. In the artificial intelligence (AI) community, this has been realized, among others, by Laird and van Lent [89]. Such environments therefore are a natural starting point for exploring a practical perspective on reasoning about knowledge in interaction, about which we give some thoughts in this chapter.

We start by briefly describing a logic framework which seems highly suitable for this endeavour.

Dynamic Epistemic Logic (DEL). DEL is a sophisticated and intuitive formalism for reasoning about epistemic situations and change. It was initiated by Gerbrandy and Groeneveld [66] and Baltag et al. [14]. The book by van Ditmarsch et al. [52] provides a recent overview.

The fundamental idea of DEL is to add a dynamic aspect to epistemic logic, which can be represented on the level of models. Besides the (main) model describing a given situation as illustrated in the Introduction chapter, DEL has action models describing actions (or events).

An action model has a similar structure as the main model, but its nodes represent possible actions, including an actual action, and its edges represent the agents’ uncertainties with respect to these possible actions. For example, if Ann sees Bob showing the ace of spades to Eve, but Ann cannot see which card it is, she will also consider it possible that he is showing her the queen of hearts (or any other card).

An action model can be applied to a model using a product update operation
which yields a new model reflecting the situation after the corresponding action has occurred. Actions which not only affect the epistemic situation but change actual facts in the world have been examined [50], but the more widely studied purely epistemic actions already suffice for many interesting scenarios, for example knowledge games as described in Chapter 4.

**Using DEL in simulated worlds.** One way to employ DEL (or logic systems in general) is through its axioms. These fundamental truths and rules can be used to deduce any valid formula. A possible approach then is to combine a virtual world that uses deductive planning, such as the one by Magnusson and Doherty [95], with axioms for epistemic reasoning. 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.

However, as throughout this dissertation, we here focus on a model-based viewpoint. In the following, we describe some issues and possibilities on the way towards realistic implementations.

There are a number of reasons why we believe that DEL, and in particular a model-based approach, is very suitable for introducing epistemic reasoning to computer-simulated worlds:

- A centralized model comprising all agents lends itself naturally to simulated worlds, where there already is a central controller providing agents with information such as current vision and other physical senses.\(^1\)

- Importantly for real-time applications, a model-based approach can allow for more efficient evaluation of epistemic formulas than a deduction-based approach.

- The DEL model is updated as events occur to reflect the current state of affairs. It does not retain information about past states, as other, for example history-based, frameworks do; this keeps the model manageable.

- Focusing on the model, rather than the axioms, is more intuitive (in particular in the case of DEL), which makes it easier to find suitable optimizations and restrictions of the logic. A model-based implementation can also be more straightforward to inspect and debug than a heap of formulas and deductions.

- The data structures for DEL models and the operations for updates are straightforwardly represented in any programming language.

\(^1\)However, the centralized viewpoint is not vital to our aim of implementing DEL, and any results and techniques obtained for a centralized model are likely transferable to autonomous agents with internal epistemic models, due to close correspondences between these viewpoints [11].
As illustrated in Chapter 4, artificial agents acting in a simulated world can then, for example, use epistemic conditions to branch over alternative program paths. Whenever an epistemic formula is encountered, the knowledge module, which maintains the DEL model, is queried to evaluate it. Thus, agents gain access to their knowledge and beliefs just like they receive information about their physical senses from the central simulation controller.

**Issues.** The biggest problem of a model-based approach is that the model to be maintained can be very large. The number of possible configurations of atoms is exponential in the number of atoms, and in order to keep track of higher-order knowledge, the same configuration of atoms may even need to be embodied in several states. In DEL, update operations exacerbate this problem further: the product operation *prima facie* increases the size of the model exponentially.

Correspondingly, implementations of DEL are rare. There exists one generic implementation of DEL models and operations (DEMO, [54]). However, without precautions the models quickly become too large to handle, and it is unclear to what extent realistic scenarios stay manageable in the long run.

Real-time simulation environments, having to simulate concurrently all parts and aspects of the world, make efficiency of both representation and processing a top priority, especially in computer games aimed at customary home computers. Thus, the crucial point is to find compact ways of representing models, examine how exactly updates affect the size of the model in the long run, and find conditions on actions and epistemic formulas under which an implementation becomes feasible. In order to obtain results that are relevant for general applications, this needs to be done in a more generic and systematic way than what we have done in Chapters 3 and 4.

In the following, we discuss some starting points for tackling these issues.

**Representation and evaluation of models.** Besides ensuring that the maintained model is always the smallest of all equivalent ones,\(^2\) the model needs to be represented in a way that can be stored efficiently. This is one of the central points in the area of model checking [41], and that area can thus serve a starting point for finding techniques for compact representation and efficient evaluation.

However, DEL brings an additional twist in the form of update operations. These should operate directly on the compact representation of the model, without having to blow up the model in some intermediate steps. Since this adds a requirement to the representation techniques, they need to be revisited and adapted where necessary.

**Restrictions on events.** One way to avoid uncontrolled exponential growth of the model as events occur is to impose restrictions on the classes of events that

---

\(^2\)This is the so-called *bisimulation contraction.*
Chapter 7. Outlook

will be taken into account in the simulation. For example, the events considered in Chapter 4 have an effect only the first time they occur. This can be seen by noting that the order in which different events occur does not matter, and that any event occurring twice in a row has the same effect as only one occurrence.

Along similar lines, more general classes of events which are commutative and idempotent in an adequate sense may be of relevance. It may also be possible to identify other technical conditions which make updates “well-behaved” in the long run. It is then necessary to examine what these conditions imply on the logical level, what configurations of knowledge they give rise to, and whether the resulting classes of events are useful in practice.

In the DEL community there is increasing interest in issues such as what events are equivalent, how updates behave in the long run, and whether models ultimately stabilize. This line of research, initiated by Ruan [128], Sadzik [131], and van Eijck et al. [55], is closely related to the aim of finding classes of events that are suitable for implementations.

Restrictions on epistemic formulas. Naturally, the class of epistemic formulas that are to be evaluated by the knowledge module has a direct impact on its efficiency. Imposing restrictions, for example, on the nesting depths of knowledge and belief operators allows to bound the evaluation time of formulas, and to prune down the model: information that is never going to be queried does not need to be modeled. Note also that there is some interplay between the different kinds of restrictions. For example, similarly to what we have seen in Chapter 2, certain restrictions on initial situations and events make certain formulas equivalent, since models discriminating them will never occur. In this sense, there are corresponding restrictions of events and formulas.

Here, too, the update operation should be specialized to these restrictions and operate directly on pruned models. That is, rather than generating a temporary model and pruning it down, it should, if possible, avoid generating unnecessary information in the first place.

In the proposed context of virtual worlds, the ultimate goal is to maximize efficiency while retaining realism, or enjoyability in the case of computer games. Therefore, as described in Chapter 4, Section 4.6.3, insights about human cognition need to be used in order to find suitable epistemic languages. Conversely, technical restrictions that allow for efficient implementations can be examined on a logical level to find possible application scenarios.

Conclusion. Overall, we believe that an implementation-oriented perspective can provide new and useful stimuli for future directions of the mentioned lines of research.