MCK: Automated Analysis of Knowledge and Probability in Multi-Agent Systems

White Paper

Executive Summary: Reasoning about dynamic systems in which one or more agents operate under conditions of incomplete information can be highly complex. Epistemic Model Checking is an approach to the automated formal analysis of such systems. The approach focuses on how the states of information of agents change over time. The epistemic model checker MCK can automatically verify whether a specifications expressed in a formal logic of knowledge, probability and time holds in such a system. This document summarizes the current capabilities of the tool, and describes some sample applications in distributed systems, security analysis and defence.

Problem: Analysis of Knowledge and Probability in a Multi-Agent System

The need to analyze problems involving agents operating with uncertainty about the state of their environment arises in many applications, such as distributed and concurrent computing, economic modelling, diagnosis, artificial intelligence, computer security and defence. In particular, in such problems it is of interest to determine how an agent's *epistemic state*, i.e., its state of information, changes over time. Such analyses can be difficult to conduct by hand, and can be quite subtle.

One simple illustration of the difficulty of such analyses is the well-known Monty Hall problem [7]. Monty Hall, a game show host, offers a contestant a choice of three doors. Behind one of the doors is a car, behind the other two is a goat. After the contestant has selected a door, Monty opens one of the other doors, to reveal a goat, and offers the contestant the opportunity to switch their choice to the third door, that remains closed. Should the contestant stay with their choice or switch? To solve the problem, we need to answer the following probabilistic question about the contestant's epistemic state:

> "After observing the goat behind the door opened by Monty, is the contestant's probability that the car is behind the door selected less than their the probability that the car is behind the remaining closed door?"

If so, it is to the contestant's advantage to switch. Note that the problem involves reasoning about a scenario in which agents perform actions in response to their observations, and the state of the system changes over time (e.g., a closed door is opened, and the agents observe what is behind it).

That such questions can be difficult to answer is shown by the fact that when presented in a Parade magazine column [7], this simple puzzle provoked a heated debate, with numerous professional mathematicians supporting the "wrong" answer that there is no advantage to switching. In fact, the correct answer depends on some missing information: is Monty's behaviour to randomly choose one of the unselected doors, or, arguably more plausibly, to always open a door with a goat behind it? It is in the latter case that there is an advantage to switching. The dependence of the answer on Monty's behaviour demonstrates the importance of a careful dynamic modelling and analysis of the problem.

Solution: Epistemic Model Checking

The difficulty of reasoning about problems concerning agents' epistemic state in dynamic settings indicates a need for automated tool support. In a program of research conducted at UNSW Sydney, with funding from the Australian Research Council, Defence R&D Canada, and the US Air Force Office of Scientific Research, a model checker MCK [4] has been developed that provides such tool support.

Model checking is a methodology for automated verification that is widely applied in computer software and hardware verification. The input to a model checker is a description of a system design (typically in the form of program code) and a question concerning this design

(typically expressed in a formal logic). The output of the model checker is an answer to that question, sometimes together with an explanation of the answer in the form of a possible system behaviour.

In the case of MCK, the system design is represented as a program, and the question is phrased in a formal language that can express properties concerning time, probability and knowledge. (An agent *knows* a fact if it is true in all the situations the agent considers to be possible, given the information it has. This gives a non-numerical notion that allows for more efficient automated analysis than is possible when dealing with numerical probabilities, but is often informative.) MCK supports several different semantics for the agent's epistemic state, depending on whether an agent determines what it knows from just its current observation, its current observation plus the current time, or its history of observations (interpreted either synchronously or asynchronously). The temporal dimension is supported by both linear and branching time temporal expressiveness. MCK implements a variety of algorithmic approaches to the model checking problem: a binary decision diagram based technique, a propositional satisfiability-based bounded model checking technique, as well as an explicit state model checker. MCK can be run via a graphical user interface as well as a command-line tool. Debugging of designs and specifications is supported by state-space visualization and a debugging game that helps the user to understand the reasons for a failed specification.

A web application allowing MCK to be trialed is available at

`http://www.cse.unsw.edu.au/~mck`.

Publications on the underlying theory, and on the applications described below, can also be found at that site.

Sample Application: Distributed Systems/Concurrent Protocol Analysis

One of the motivations for the application of epistemic notions in computer systems design is the that in the protocols used in concurrent and distributed systems, the designer needs to reason about what information about the global state of the system is encoded in the local state of one of the agents operating in the system. It is often cheaper to compute from local information, when that suffices for a decision to be taken, than to expend communication costs and time in gathering information. Thus, it is useful to a designer to know under what circumstances the information required to take a decision is available locally.

One example [2] demonstrating the usefulness of MCK in analysis of this sort of question is in multiprocessor cache coherency protocols. These aim to optimize computation times in the processors by means of caches (local memories that operate faster than main memory). They need to deal with the complex issue of how to maintain consistency when multiple processors are using the same data from main memory. Here the global information is "is the data currently in a cache line equal to the correct value?" If so, this data can be used without expending time and communication costs to fetch data from main memory. We modelled several classical cache coherency protocols, and MCK was able to automatically discover a behaviour in one of these protocols where the information that data in a cache was valid could have been deduced from the local state, but the protocol nevertheless redundantly fetched a copy from main memory. Elimination of this redundancy leads to an optimization of the protocol.

This example demonstrates that MCK has the potential to be a useful tool for such protocol optimization problems. There is also significant potential for applications in fault-tolerant protocol design, where the reasoning about what information is encoded in local state can be significantly more subtle.

Sample Application: Computer Security Analysis

Reasoning about epistemic state of agents is also critical in analysis of computer security, where notions such as secrecy and anonymity are concerned what an adversary does not know. We have applied MCK to a number of security protocol analysis problems, where we have demonstrated its usefulness in automatically verifying whether such properties hold.

For example, in [1], we used MCK to discover a subtle flaw in a protocol for anonymous broadcast proposed by Chaum [3]. The protocol aims to ensure that a message is received by all agents, without the sender's identity being revealed. Our automated analysis led to the discovery of a behaviour of the protocol, where, although the sender's identity remains secret, some of the bits of a message are deducible by only a subset of the intended recipients, rather than all.

In another study (unpublished) we considered a proposed protocol [8] that seeks to defend against cache timing attacks. Such attacks have been shown to allow secret keys to be deduced by an attacker that shares a cache with a cryptographic computation, even when no main memory lines are shared. The defence aims to block this attack by randomizing the cache layout. We are able to automatically identify a weakness in this proposed defence.

Sample Application: Pursuit-Evasion Problems

Region clearing is an instance of a *pursuit-evasion* problem of interest in defence and security applications. Consider a building to be closed at night: we need to ensure that no occupants or intruders remain in the building. One or more guards are to be sent through the building to search for such occupants. Given a strategy for movement of the guards through the building, we would like to determine whether the strategy is effective, e.g., do the guards know, after following this strategy, that the building is empty? The difficulty of the problem arises from the fact that the guards have limited visibility and any occupants may also move around the building during the search, possibly evading detection.

In a joint project with Defence R&D Canada, we applied MCK to the analysis of such scenarios [5]. The region to be cleared is modelled as a discrete graph, rendering the problem finite state. Several epistemic specifications have been identified as of interest in this setting, and it has been demonstrated that MCK is capable of automated analysis of modest scale graphs and search lengths. (Graphs of up to 50 nodes and search lengths up to 100 were solved. This is not insignificant, since the problem being solved is NP-complete.) Moreover, we have demonstrated that the problem of *finding* an optimal search strategy for a given number of guards can be solved using MCK [6]. Heuristic approaches specifically tailored to this problem, are admittedly more efficient, but may find only an approximation to the best solution. Our approach has the advantage of proceeding from a more general problem description, and both discovering a

solution and providing a guarantee of its optimality. The generality of the approach also allows us to handle novel questions without needing to develop specialised algorithms. An example of this is the question "Does this pursuit plan ensure that either an evader is eventually caught, or that the pursuers eventually know that the evader has a strategy that forever avoids capture." (Either condition could be grounds for terminating the chase.)

Contact

For more information on MCK, contact Prof. Ron van der Meyden (meyden@cse.unsw.edu.au).

References

[1] O. Al-Bataineh and R. van der Meyden. Epistemic model checking for knowledge-based program implementation: an application to anonymous broadcast. In *International ICST Conference on Security and Privacy in Communication Networks, SecureComm'10*, pages 429–447, 2010.

[2] K. Baukus and R. van der Meyden. A knowledge based analysis of cache coherence. In Jim Davies, Wolfram Schulte, and Michael Barnett, editors, *ICFEM*, volume 3308 of *LNCS*, pages 99–114. Springer, 2004.

[3] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, pages 65–75, 1988.

[4] P. Gammie and R. van der Meyden. MCK: model checking the logic of knowledge. In Rajeev Alur and Doron Peled, editors, *CAV*, volume 3114 of *LNCS*, pages 479–483. Springer, 2004.

[5] X. Huang, P. Maupin, and R. van der Meyden. Model checking knowledge in pursuit-evasion games. In *International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 240–245, 2011.

[6] X. Huang and R. van der Meyden. Synthesizing strategies for epistemic goals by epistemic model checking: an application to pursuit-evasion games. In *Proc. AAAI*, 2012.

[7] M. vos Savant. Ask Marilyn. *Parade Magazine*, page 15, Sept. 9, 1990. There were also followup articles in *Parade Magazine* on Dec. 2, 1990 (p. 25) and Feb. 17, 1991 (p. 12).

[8] Z. Wang and R. B. Lee. New cache designs for thwarting software cache-based side channel attacks. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pages 494–505, New York, NY, USA, 2007. ACM.