MCK: Model Checking the Logic of Knowledge*

Peter Gammie[1] and Ron van der Meyden[2]

[1] Computing Science, Chalmers Institute of Technology, Sweden,
peteg@cs.chalmers.se
[2] School of Computer Science and Engineering, University of New South Wales,
and National ICT Australia, Sydney, Australia
meyden@cse.unsw.edu.au

Introduction: The specification formalism employed in model checking is usually some flavour of temporal or process algebraic language that expresses properties of the *behavioural* aspects of a system. *Knowledge* [5] is a modality that is orthogonal to the behavioural dimension, capturing properties of *information flow.* Logics of knowledge have been shown to be a useful framework for the analysis of distributed algorithms and security protocols, and model checking of these logics was first mooted by Halpern and Vardi [6]. Since that time theoretical aspects of model checking the logic of knowledge and its combinations with temporal logic have been studied [8–10]. The system MCK introduced in this paper implements parts of this theory.

The Model Checking Scenario: The typical scenario that can be analysed using the system consists of some number of *agents* (which might be players in a game, actors in an economic setting, or processes, programs or components in a computational setting) interacting in the context of an *environment.* The agents have the capacity to perform certain *actions* in this environment, which they choose according to their individual *protocols*, or sets of rules. The agents have *incomplete information* about the state of the system due to the fact that they are able to *observe* only part of the state at each instant of time.

The MCK system can be used to analyse this type of setting by the use of *model checking* techniques. The input to the MCK system describes: (1) the environment in which the agents operate, including a formal description of agent names, states, initial states, actions and how they affect states, and fairness conditions; (2) the protocol for each of the named agents, and a description of what parts of the state can be observed by which agents; (3) a number of *specification formulas* to be model checked, expressing how the agents' knowledge evolves over time. Both the possible state changes selected by the environment and the agents' choices of action may be non-deterministic.

The MCK system supports several different types of temporal and epistemic specifications. In the epistemic dimension, agents may use their observations in a variety of ways to determine what they know. In the *observational* interpretation of knowledge, agents make inferences about the actual state based just

* To appear CAV 2004, copyright Springer Verlag. Work of the first author conducted while employed at UNSW. Work funded by an Australian Research Council Discovery grant. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

on their current observation. In the *clock* interpretation of knowledge, agents compute knowledge using both their current observation and the current global clock value. Even more information can be extracted by the agent if it uses a complete record of all its observations to date to determine what it knows – this is called the *synchronous perfect recall* interpretation of knowledge. In the temporal dimension, the specification formulas may use either linear time temporal logic, LTL, or the branching time logic CTL. The system supports different combinations of these parameters to different degrees, which in some cases is because the implementation remains to be undertaken and in others is due to inherent computational difficulties. In the case of the observational semantics, for example, the system supports arbitrarily nested combinations of either CTL or LTL operators with operators for knowledge and common knowledge, a type of fixpoint of the knowledge of many agents. In contrast, the perfect recall semantics only supports a linear or branching "next time" operator, although the theory for the full combination of knowledge with LTL has been developed [9].

The custom modelling language used in MCK is designed to cater for some specific issues arising from the semantics of knowledge and to allow maximum modelling flexibility; the intention is to support experimentation in this comparatively unexplored area rather than verification for a specific language.

Application Example: Figure 1 presents an example input file to the system which models a scenario where the single agent, a robot called `Robot` (running the protocol `"robot"`) operates in an environment of 8 possible positions. The environment provides noisy readings of the position to the robot via the `sensor` variable, which is declared to be an observable input to the agent. The robot's goal is to halt in the region $\{2..4\}$, which its protocol attempts to achieve by halting when the sensor value is ≥ 3. We would like to verify that this is the best the robot can do given its observations, which is to say that the condition `sensor` ≥ 3 characterises all situations in which the agent knows that it is in the goal region when it is running this particular protocol.

The `transitions` section represents the effects of the robot's actions (`Halt` and `skip`) on the state, by means of a program using non-deterministic `if` statements. This program is considered to run atomically in a single tick of the clock.

The example contains two specification formulas. The construct `spec_obs_ltl` indicates that the formula uses linear time temporal logic operators and that the knowledge operator `Knows` is to be interpreted using the observational semantics. The operator `G` expresses truth at all future times. The model checker verifies that this formula is indeed true. The use of `spec_spr_xn` in the second specification indicates that the knowledge modality should be interpreted using synchronous perfect recall, and that the formula has the form $\mathtt{X}^n\phi$, where ϕ is atemporal, expressing that ϕ holds in precisely n steps after an initial state ($n = 2$ in this example). The model checker verifies that this formula is false. Intuitively, the determinacy of the robot's motion allows it to derive its location from the number of observations made. This means that the test `sensor >= 3` is not complete — the right-to-left implication fails. If we alter the `transitions` part to allow

position to either increment or remain static and add a fairness condition to ensure progress, this formula becomes true, but is still false for $n = 3$.

```
type Pos = {0..7}
position : Pos
sensor : Pos
halted : Bool

init_cond = position == 0 /\ sensor == 0 /\ neg halted

agent Robot "robot" ( sensor )

transitions
begin
  if Robot.Halt -> halted := True fi;
  if neg halted -> position := position + 1 fi;
  if True -> sensor := position - 1
  [] True -> sensor := position
  [] True -> sensor := position + 1
  fi
end

spec_obs_ltl = G (sensor >= 3 <=> Knows Robot position in {2..4})
spec_spr_xn = X 2 (sensor >= 3 <=> Knows Robot position in {2..4})

protocol "robot" (sensor : observable Pos)
begin
  do neg (sensor >= 3) -> skip
  [] break             -> <<Halt>>
  od
end
```

Fig. 1. A Sample MCK input file.

Implementation Sketch: The system constructs a BDD representation of a data structure that encodes the knowledge set of an agent, i.e. the set of states that are consistent with its local state. For the observational and clock semantics this structure is simply a set of states, but the synchronous perfect recall semantics requires a function that maps a fixed, finite sequence of observations to the set of final states of traces consistent with this sequence of observations, as detailed in [10]. The system is implemented in Haskell, and relies on David Long's BDD package, implemented in C, for efficient BDD operations.

Related Work: While other systems in recent years have implemented model checking for the logic of knowledge, MCK is unique in its support of a range of knowledge semantics and choices of temporal language. Additionally, fairness constraints and the common knowledge operator have not been treated in prior work, and the clock semantics has not previously been implemented.

The theoretical basis for MCK is largely described in [8–10], which employs LTL as the temporal language. Theory for model checking a combination of the perfect recall semantics and CTL is discussed in [2, 11].

Wooldridge and van der Hoek [7] use a connection between knowledge and the notion of *local proposition*, introduced in [3], to reduce model checking knowledge with respect to the observational semantics to temporal model checking in SPIN. This reduction works for positive occurrences of knowledge operators but leads to an explosion in the number of temporal formulas that need to be checked when there are negative occurrences. MCK handles negative occurrences directly without incurring such an explosion.

Our work is motivated more by issues of information flow in distributed systems than by distributed artificial intelligence concerns. Other work in the Distributed AI literature on model checking multi-agent systems, such as [1, 4], use epistemic modalities such as "belief" that are not given an information theoretic semantics as in Halpern and Moses [5]. The same remark applies to related work in the literature on cryptographic protocol verification.

Conclusion: MCK can be downloaded at `http://www.cse.unsw.edu.au/~mck`. The system is under active development and other instances of the known algorithms for model checking knowledge and time will be added in due course.

References

1. M. Benerecetti, F. Giunchiglia, and L. Serafini. Multiagent model checking. *Journal of Logic and Computation*, 8(3):401–423, Aug 1997.
2. P. Bertoli, A. Cimatti, M. Pistore, and P. Taverso. Plan validation for extended goals under partial observability (preliminary report). In *AIPS 2002 Workshop on Planning via Model Checking*, Toulouse, France, April 2002.
3. K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In Itzhak Gilboa, editor, *Theoretical Aspects of Rationality and Knowledge*, pages 29–41. Morgan Kaufmann,98.
4. R. H. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *CAV*, pages 110–113, 2003.
5. J. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.
6. J. Halpern and M. Y. Vardi. Model checking vs. theorem proving: A manifesto. Technical Report RJ 7963, IBM Almaden Research Center, 1991.
7. W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *9th Workshop on SPIN (Model Checking Software)*, Grenoble, April 2002.
8. R. van der Meyden. Common knowledge and update in finite environments. *Information and Computation*, 140(2), 1998.
9. R. van der Meyden and N.S. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. Conf. on Software Technology and Theoretical Computer Science*, Springer LNCS No 1738, pages 262–273, Chennai, 1999.
10. R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *IEEE Computer Security Foundations Workshop*, 2004.
11. N. Y. Shilov and N.O. Garanina. Model checking knowledge and fixpoints. In *FLOC workshop on Fixed Points in Computer Science*, 2002.