# A Framework for Model Checking against CTLK Using Quantified Boolean Formulas[⋆]

Emily Yu, Martina Seidl, and Armin Biere

Institute for Formal Models and Verification,
Johannes Kepler University Linz, Austria
{zhengqi.yu, martina.seidl, biere}@jku.at

**Abstract.** We present a novel bounded model checking (BMC) tool chain for multi-agent systems. This framework automatically translates the verification of system models against properties formulated in computation tree logics with epistemic modalities (CTLK) into quantified Boolean formulas (QBFs). Our framework exploits recent QBF technology for solving those verification problems and for certifying the result, making the implementation of a dedicated CTLK solver obsolete. The translation to QBF is based on existing theoretical work and implemented in our novel tool $\mathsf{MCMAS}_{\mathsf{qbf}}$ which extends the open-source model checker MCMAS. First experimental results are very promising and indicate the practical feasibility of our approach. Furthermore we provide novel benchmarks to the QBF community.

**Keywords:** Bounded Model Checking · QBFs · Multi-agent Systems

## 1 Introduction

Multi-agent systems (MAS) are nowadays applied in various fields to describe complex systems. For example, MAS are used to formalize the interactions of different components that act independently [8]. To verify their correctness, *Computation Tree Logic* with knowledge (CTLK) has been introduced [10]. Besides temporal operators like "**A**lways", "**U**ntil", and "**F**inally", CTLK also includes formulas with knowledge modalities $\mathbf{K}_i\phi$ expressing that "Agent $i$ knows $\phi$".

With CTLK it becomes possible to perform model checking for MAS [13]. *Model Checking* [1, 5, 6] is an important technique for verifying safety-critical systems against properties expressed in temporal logics like LTL or CTL. To deal with the so-called *state explosion problem* of model checking, SAT-based bounded model checking (BMC) [3] was introduced. To obtain more compact encodings of BMC problems than possible with SAT, encodings of BMC to *quantified Boolean formulas* (QBFs) have been presented [7]. Such encodings exploit the power of existential and universal quantifiers to avoid duplications of formula parts.

[⋆] This work was supported by by the Austrian FWF grant W1255-N23 and the LIT AI Lab funded by the State of Upper Austria.

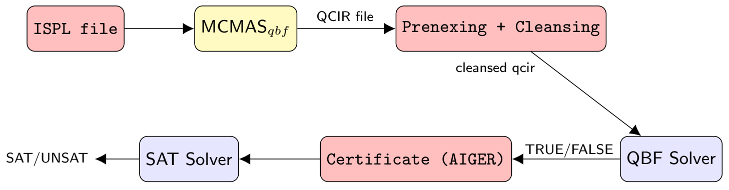

Fig. 1: The complete MCMAS$_{\mathsf{qbf}}$ tool chain.

In this paper, we present a fully automatic tool chain for verifying descriptions of multi-agent systems against properties in CTLK. Therefore, we implemented MCMAS$_{\mathsf{qbf}}$ for translating such BMC problems to QBFs building upon the bounded semantics of CTLK introduced in [16].

## 2 The MCMAS$_{qbf}$ Tool Chain

Our tool MCMAS$_{\mathsf{qbf}}$ [14] extends MCMAS [12], an open-source model checker for the verification of multi-agent systems supporting various temporal epistemic logics. We reused the parser of MCMAS to obtain the interpreted system data structures based on which we generate the QBF encodings. We implemented the translation of bounded semantics of CTLK into QBFs based on the theoretical work in [16] which includes both existential and universal fragments of the logic. As an approximation to unbounded model checking, the bounded semantics considers a finite state space where each path in the system is restricted to a length of $k$. However, in the verification process the search space is extended progressively as the formula is evaluated.

The input of MCMAS$_{\mathsf{qbf}}$ is an ISPL file which contains a description of the system and a CTLK formula for the property to be checked. ISPL is an agent-based, modular language based on the interpreted systems [9] formalism commonly used for MAS. Our extension is invoked with parameters

```
-QBFbmc [k] [QCIR-File] [ISPL-File],
```

where $k$ is a value specifying the bound followed by an ISPL file and a QCIR output file for the QBF. Our tool MCMAS$_{\mathsf{qbf}}$ is embedded in the tool chain as shown in Figure 1. It produces QBFs in the most general variant of the QCIR format [11], i.e., in non-prenex form which allows to position quantifiers arbitrarily within a formula. Since there is no state-of-the-art QBF solver that supports this general format, an additional prenexing step is necessary to shift the quantifiers to the front. For example, the formula $\forall x \exists y \phi \wedge \forall a \exists b \psi$ has to be rewritten to $\forall a, x \exists b, y(\phi \wedge \psi)$. Therefore, we implemented a simple tool that performs not only quantifier shifting, but also the translation to the cleansed QCIR format that requires the names of the Boolean variables to be numbers and not strings. Now the QBF can be passed to any QBF solver that is able to process formulas in the cleansed QCIR format. We applied the Quabs [15] that can not only decide the truth value of the formula but also produce certificates.

These certificates are And-Inverter-Graphs (AIGs) [4] representing the solution to the BMC problem, and can be checked by a SAT solver for increasing trust in the QBF solver. For this purpose, we use the SAT solver PicoSAT [2].

## 3 Case Study

As a case study, we consider the popular Train-Gate-Controller (TGC) example [10]. In this scenario, there are multiple trains on different tracks and a controller. The tracks intersect at one tunnel which has red-green lights controlled by the controller, and only one train can operate in the tunnel at a time when the light is green. The following code snippet describes this scenario for one train modeled in ISPL:

```
Agent train1
Vars:
state: {wait, tunnel, away};
end Vars
Actions = {enter, leave, nothing};
Protocol:
state = wait: {enter, nothing};
state = tunnel: {leave, nothing};
state = away: {nothing};
end Protocol
Evolution:
state = wait if state = away and Action = nothing;
state = tunnel if state = wait and Action = enter and Environment.Action=enter1;
state = away if (state = tunnel and Action = leave and Environment.Action=leave1)
or (state=wait and Action=nothing);
end Evolution
end Agent
```

An interpreted system typically contains a set of agents (`train1`, ...) with possible local states (`wait`, `tunnel`, `away`), actions (`enter`, `leave`, `nothing`), as well as protocols and evolution functions for describing the system behavior. The global states are composed of each agent's local states. Further an initial state is also defined in the ISPL description. To translate the model checking problem into QBFs, we firstly need to encode the interpreted system as follows:

- *state space*: $\lceil \log |L_i| \rceil$ Boolean variables are needed for representing the local states $L_i$ of agent $i$. The same number of variables is needed for the local successor state. The global current state $\mathbf{v} = (v_e, v_1, ..., v_N)$ and the global successor state $\mathbf{v}' = (v'_e, v'_1, ..., v'_N)$ are vectors of local states where $N$ is the number of agents and $e$ refers to the environment.
- *actions*: For the actions, $\sum_{i \in \{e,1..,N\}} \lceil \log |Act_i| \rceil$ Boolean variables are needed.
- *transition relation*: For each agent, the protocol function and evolution function are encoded symbolically using $v_i$ and $v'_i$. The global transition relation is the composition of protocol and evolution functions based on $\mathbf{v}$ and $\mathbf{v}'$.

We have implemented the encoding presented in [16] and our implementation allows to generate a QBF as a QCIR file which then can be solved and certified by existing QBF solvers. The property holds if the verification result of the QBF solver shows the formula is satisfied, and vice versa.

Table 1: Experimental results obtained for the Train-Gate-Controller case study.

| $N$ | $k$ | $\phi_{qbf}$ (gates) | $C$ | $t_{total}$(s) | $t_{qs}$(s) | $t_{sat}$(s) |
|---|---|---|---|---|---|---|
| 3 | 5 | 30616 | 990 | 0.961 | 0.056 | 0.023 |
| 3 | 10 | 110971 | 2945 | 3.445 | 0.222 | 0.088 |
| 3 | 15 | 252226 | 5950 | 8.018 | 0.673 | 0.201 |
| 3 | 20 | 464881 | 10005 | 14.942 | 1.178 | 0.370 |
| 5 | 5 | 57695 | 2354 | 1.785 | 0.112 | 0.050 |
| 5 | 10 | 206735 | 7299 | 6.469 | 0.487 | 0.156 |
| 5 | 15 | 464875 | 14994 | 15.057 | 1.261 | 0.359 |
| 5 | 20 | 848615 | 25439 | 28.560 | 2.820 | 0.685 |
| 8 | 5 | 100482 | 5480 | 3.127 | 0.226 | 0.084 |
| 8 | 10 | 357472 | 17460 | 11.299 | 0.982 | 0.280 |
| 8 | 15 | 798762 | 36240 | 26.195 | 2.794 | 0.627 |
| 8 | 20 | 1449852 | 61820 | 51.409 | 6.882 | 1.259 |
| 10 | 5 | 140217 | 9747 | 4.607 | 0.557 | 0.114 |
| 10 | 10 | 495812 | 30452 | 16.771 | 2.247 | 0.389 |
| 10 | 15 | 1101507 | 62707 | 38.680 | 6.073 | 0.871 |
| 10 | 20 | 1988802 | 106512 | 76.313 | 14.219 | 1.578 |

We verify the following property in our case study: along all paths in the system, it is always the case that if train$_1$ is in the tunnel then it knows that the other trains cannot be operating in the tunnel at the same time. In CTLK, this property can be expressed as follows:

$$\phi = \mathbf{AG}\ (\,\text{in\_tunnel}_1\ \rightarrow\ \mathbf{K}_{\text{train}_1} \bigvee_{i=2}^{N} \neg\text{in\_tunnel}_i)$$

To evaluate the performance of the tool chain, we ran several experiments on an Intel Core™ i7-2600 machine with 3.40GHz CPU and 16GB RAM running Ubuntu v18.04.2 (Linux kernel v4.15). We evaluated the bounded model checking problem with different values of $k$, and in order to test the scalability of the framework, we ran experiments with $5, 8$, and 10 trains (we use $N$ to represent the number of trains).

Table 1 reports the results of our case study. The obtained cleansed QBFs in prenex form contain up to $2M$ gates in the QCIR format, while the certificates in AIGER contain only up to $100K$ gates plus the inputs and outputs related to the QBF variables ($C$ in Table 1). The solving time $t_{total}$ includes the time for the whole tool chain including the QBF solving time $t_{qs}$ and the time for checking the certificates $t_{sat}$. While the solving times are quite small, much time is needed for the encoding and the cleansing. Here, many optimizations are possible.

## 4 Discussion

We presented a complete tool chain for solving bounded model checking of multi-agent systems against CTLK specifications using QBF solving technology. First experiments are very promising, allowing us not only to solve the BMC problems but also to obtain quite small certificates from the QBF solvers. Further, this work provides practical benchmarks in the general QCIR format to the QBF community.

As future work, we plan to integrate the model checker with a QBF solver more tightly using an incremental QBF approach to speed up model checking. The translation algorithm can also be optimized further, by for instance picking an arbitrary value of $k$ as a starting point and increase $k$ step-wise in a loop. Furthermore, sub-formulas can be encoded separately then verified, and the verification results can be cached in order to speed up the whole model checking process when applied in a real-world setting.

## References

1. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
2. Biere, A.: Lingeling, Plingeling, PicoSAT and PrecoSAT at SAT Race 2010. Tech. rep., FMV Reports Series, Inst. FMV, JKU Linz, Austria (2010)
3. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without bdds. In: Proc. of TACAS'99. LNCS, vol. 1579, pp. 193–207. Springer
4. Biere, A., Heljanko, K., Wieringa, S.: AIGER 1.9 and beyond. Tech. rep., FMV Reports Series, Inst. FMV, JKU Linz, Austria (2011)
5. Clarke, E.M., Grumberg, O., Kroening, D., Peled, D., Veith, H.: Model Checking. MIT press (2018)
6. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R. (eds.): Handbook of Model Checking. Springer (2018)
7. Dershowitz, N., Hanna, Z., Katz, J.: Bounded model checking with QBF. In: Proc. of SAT'05. LNCS, vol. 3569, pp. 408–414. Springer (2005)
8. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: A survey. IEEE Access **6**, 28573–28593 (2018)
9. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press, Cambridge, MA, USA (2003)
10. van der Hoek, W., Wooldridge, M.J.: Tractable multiagent planning for epistemic goals. In: AAMAS. pp. 1167–1174. ACM (2002)
11. Jordan, C., Klieber, W., Seidl, M.: Non-cnf QBF solving with QCIR. In: AAAI Workshop: Beyond NP. AAAI Workshops, vol. WS-16-05. AAAI Press (2016)
12. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multi-agent systems. In: CAV. LNCS, vol. 5643, pp. 682–688. Springer (2009)
13. Lomuscio, A., Raimondi, F.: The complexity of model checking concurrent programs against CTLK specifications. In: DALT. LNCS, vol. 4327, pp. 29–42. Springer (2006)
14. MCMAS-QBF: (2019), http://fmv.jku.at/ftscs19
15. Tentrup, L.: Non-prenex QBF solving using abstraction. In: Proc. of SAT'16. LNCS, vol. 9710, pp. 393–401. Springer (2016)
16. Zhou, C., Chen, Z., Tao, Z.: QBF-based symbolic model checking for knowledge and time. In: TAMC. LNCS, vol. 4484, pp. 386–397. Springer (2007)