Abstraction for Model Checking Modular Interpreted Systems over ATL*

(Extended Abstract)

Michael Köster
Computational Intelligence Group
Clausthal University of Technology
michael.koester@tu-clausthal.de

Peter Lohmann
Theoretical Computer Science
Leibniz University Hannover
lohmann@thi.uni-hannover.de

ABSTRACT

We propose an abstraction technique for model checking multi-agent systems given as modular interpreted systems (MIS) which allow for succinct representations of compositional systems. Specifications are given as arbitrary ATL formulae, i.e., we can reason about strategic abilities of groups of agents. Our technique is based on collapsing each agent's local state space with hand-crafted equivalence relations, one per strategic modality. We develop a model checking algorithm and prove its soundness. This makes it possible to perform model checking on abstractions (which are much smaller in size) rather than on the concrete system which is usually too complex, thereby saving space and time.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; D.2.4 [**Software Engineering**]: Software/Program Verification—*Model checking*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Temporal logic*

General Terms

Theory, Verification

Keywords

model checking, abstraction, temporal and strategic logics, modular interpreted systems

1. INTRODUCTION

While an important feature of a Multi-agent system (MAS) is its modularity, only a few of the existing compact representations are modular, computationally grounded [15] and allow to represent knowledge and strategic ability. Among these few approaches are Modular Interpreted Systems (MIS) [11] which we use to apply our abstraction techniques. But certainly our techniques could be used with other formalisms as well. MIS are inspired by interpreted systems [8, 7] but achieve a modularity and compactness property much like concurrent programs [13], i.e., they are modular, compact and computationally grounded while allowing at the same time to represent strategic abilities. Modelling side effects of actions on states of other agents, however, is difficult to model in the latter – that is why we use MIS.

A major obstacle to model checking real systems is the state explosion problem. As algorithms require a search through the state space of the system, the efficiency of any algorithm highly depends on the size of this state space. We therefore need to eliminate irrelevant states by using appropriate abstraction techniques [2] which guarantee that the property to be verified holds in the original system if it holds in the abstract system. Hence, we reduce the local state space of each agent in a MIS by using hand-crafted equivalence relations. They are hand-crafted since any automatic abstraction generation or refinement (as in [9] for two-player games) can only work in typical cases but not in the worst case.

While abstraction of reactive systems for temporal properties is a lively research area [1, 4, 14], there are only a few approaches when it comes to MAS and even fewer concerning an abstraction technique for dealing with strategic abilities (cf. [3, 5, 6, 10]). The technique in [10] is quite similar to ours but still more restricted in an important way. They assume that there are only two agents present and then use a single abstraction to model check the whole formula. Our approach allows for multiple agents and for many abstractions (one per strategic operator). Thus we allow for a much finer control over what information is abstracted away but still preserve soundness of our model checking algorithm.

2. MIS AND ATL

We model a MAS as MIS: Each agent is described by a set of possible local states and a function that calculates the available actions in a certain state. A local transition function specifies how an agent evolves from one local state to another. States are labeled with a set of propositional symbols by an associated labeling function. Finally, an agent is equipped with a function that defines the possible influences of an agent's action on its environment, i.e., the other agents, and a function for the influence of the environment on this particular agent. We can now specify strategic properties using this framework together with ATL.

Cite as: Abstraction for Model Checking Modular Interpreted Systems over ATL (Extended Abstract), Michael Köster and Peter Lohmann, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

3. ABSTRACTION FOR MIS

In general, multi-agent systems have large associated state spaces and even if they are symbolically represented it is infeasible to verify properties by considering *all* reachable states. Nevertheless, interesting properties often only refer to parts of a system. Because of that we reduce the state space by removing and/or combining irrelevant states. Due to the modularity of MIS, we can firstly remove the obviously non-relevant parts of the global state space by removing irrelevant agents. Secondly, we reduce the state space of each agent by abstraction. As in [2, 3] we do this by partitioning the state space into equivalence classes. Each class collects all concrete states that are equivalent and forms one new abstract state. This new state is labeled by those propositions which are shared by all concrete states. We define the local transition functions of the abstract system in such a way that it behaves as the concrete one. The set of available actions in an abstract state is decreased for some agents, and increased for the rest, so that it contains exactly all actions available in every one, respectively any, of the equivalent concrete states.

4. THE MODEL CHECKING ALGORITHM

Our algorithm takes as input a MIS S, a set *init* of global states of S (the initial states), an ATL formula φ and for each strategic operator in φ, i.e., each quantified subformula ψ of φ, an abstraction relation $\equiv_\psi$. It either returns true or it returns unknown but it will never return false. If it returns true it is guaranteed that $S, q \models \varphi$ for all $q \in init$. But if it returns unknown we do not know whether S satisfies φ or not. The algorithm runs in time

$$O\left(|init| + |S| \cdot |\varphi|\right) \cdot 2^{O\left(\sum_{\psi \in \mathrm{qsf}(\varphi)} \left|S^{[\![\psi]\!]}_{\equiv_\psi}\right|\right)}$$

where $|S|$ denotes the size of the MIS S in a compact representation. The cardinality of the global state space of S may then be upto $2^{\Theta(|S|)}$. And the above algorithm is sound, i.e., if it outputs true then $S, q \models \varphi$ for all $q \in init$.

5. CONCLUSION

In this extended abstract we presented a technique to cope with the state explosion problem. That opens the path to reducing the state space of a MAS so that model checking might become tractable. Clearly, there cannot be a generic automatizable abstraction technique since model checking ATL for MIS is *EXPTIME*-complete. Hence, there are instances for which no abstraction technique at all is applicable. Consequently we focused on hand-crafted abstraction relations and proved that the presented model checking algorithm is sound, i.e., if the algorithm claims that a property holds then it really does. Of course, using hand-crafted abstraction always leads to losing completeness.

Defining different abstraction relations for each quantifier allows to shrink the state space for each subformula. We decided to take MIS as the modelling framework and argued that for any framework the modularity is important not only because of the nature of MAS but also due to the ability of reducing the state space by removing agents that are not necessary when checking a certain property. We therefore introduced a modified version of a MIS and defined an abstraction over it. For a full description of our approach see [12].

This work was funded by the NTH Focused Research School for IT Ecosystems. NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.

6. REFERENCES

[1] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[2] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Trans. Program. Lang. Syst.*, 16(5):1512–1542, 1994.

[3] M. Cohen, M. Dam, A. Lomuscio, and F. Russo. Abstraction in model checking multi-agent systems. In *AAMAS (2)*, pages 945–952, 2009.

[4] S. Das and D. Dill. Successive approximation of abstract transition relations. In *Logic in Computer Science, 2001. Proceedings. 16th Annual IEEE Symposium on*, pages 51–58. IEEE, 2002.

[5] F. Dechesne, S. Orzan, and Y. Wang. Refinement of kripke models for dynamics. In J. Fitzgerald, A. Haxthausen, and H. Yenigun, editors, *Theoretical Aspects of Computing - ICTAC 2008*, volume 5160 of *Lecture Notes in Computer Science*, pages 111–125. Springer Berlin / Heidelberg, 2008.

[6] C. Enea and C. Dima. Abstractions of multi-agent systems. In H.-D. Burkhard, G. Lindemann, R. Verbrugge, and L. Varga, editors, *Multi-Agent Systems and Applications V*, volume 4696 of *Lecture Notes in Computer Science*, pages 11–21. Springer Berlin / Heidelberg, 2007.

[7] J. Halpern, R. Fagin, Y. Moses, and M. Vardi. Reasoning about knowledge. *Handbook of Logic in Artificial Intelligence and Logic Programming*, 4, 1995.

[8] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3:159–177, 1989.

[9] T. A. Henzinger, R. Jhala, and R. Majumdar. Counterexample-guided control. In *ICALP*, volume 2719 of *Lecture Notes in Computer Science*, pages 886–902. Springer-Verlag, 2003.

[10] T. A. Henzinger, R. Majumdar, F. Y. C. Mang, and J.-F. Raskin. Abstract interpretation of game properties. In *Proceedings of the 7th International Symposium on Static Analysis*, SAS '00, pages 220–239, London, UK, 2000. Springer-Verlag.

[11] W. Jamroga and T. Ågotnes. Modular interpreted systems. In E. H. Durfee, M. Yokoo, M. N. Huhns, and O. Shehory, editors, *Proceedings of AAMAS'07*, pages 892–899. ACM Press, 2007.

[12] M. Köster and P. Lohmann. Abstraction for Model Checking Modular Interpreted Systems over ATL. Technical Report IfI-10-13, Clausthal University of Technology, 2010.

[13] O. Kupferman and M. Y. Vardi. An automata-theoretic approach to modular model checking. *ACM Trans. Program. Lang. Syst.*, 22:87–128, January 2000.

[14] R. Kurshan. *Computer-aided verification of coordinating processes: the automata-theoretic*

approach. Princeton Univ Press, 1994.

[15] M. Wooldridge. Computationally grounded theories of agency. *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS 2000)*, 2000.