TU Clausthal

Clausthal University of Technology

Abstraction for Model Checking Modular Interpreted Systems over ATL

Michael Köster and Peter Lohmann

IfI Technical Report Series IfI-10-13

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Federico Schlesinger
Contact: federico.schlesinger@tu-clausthal.de
URL: http://www.in.tu-clausthal.de/forschung/technical-reports/
ISSN: 1860-8477

The IfI Review Board

Abstraction for Model Checking Modular Interpreted Systems over ATL

Michael Köster and Peter Lohmann

Computational Intelligence Group, Clausthal University of Technology
Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany
mko@tu-clausthal.de

Theoretical Computer Science, Leibniz University Hannover
Appelstr. 4, 30167 Hannover, Germany
lohmann@thi.uni-hannover.de

Abstract

We present an abstraction technique for model checking multi-agent systems given as modular interpreted systems (MIS) (introduced by Jamroga and Ågotnes). MIS allow for succinct representations of compositional systems, they permit agents to be removed, added or replaced and they are modular by facilitating control over the amount of interaction. Specifications are given as arbitrary ATL formulae: We can therefore reason about strategic abilities of groups of agents.
Our technique is based on collapsing each agent's local state space with handcrafted equivalence relations, one per strategic modality. We present 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. We illustrate our technique with an example in a scenario of autonomous agents exchanging information.

1 Introduction

Multi-agent systems (MAS) and their logical frameworks have attracted some attention in the last decade. Agent logics have been used to reason about knowledge, time, strategic abilities, coordination and cooperation [13, 10, 1]. An important technique for verifying properties of a system is model checking [5], which has been refined and improved over the last years.

While an important feature of a MAS is its modularity, e.g., removing, replacing, or adding an agent, only a few of the existing compact representations are both modular, computationally grounded [22] and allow the system designer to represent knowledge and strategic ability. Among these few

approaches are Modular Interpreted Systems (MIS) [17], which we modify a bit, and use it to apply our abstraction techniques. MIS are inspired by interpreted systems [12, 13] but achieve a modularity and compactness property much like concurrent programs [18], i.e., they are modular, compact and computationally grounded while allowing at the same time to represent strategic abilities.

Although explicit models (and symbolic representations [21, 20]) achieve the second part very well (because the semantics are defined over them) some problems arise with the first part: Usually temporal models have an exponential number of states and, in addition, they do not support modularity since there is no easy way to remove or replace an agent. Interpreted systems [12, 13], however, have a modular state space. But they use a joint transition function for modelling temporal aspects of the system and are thus not modular wrt actions. In contrast, concurrent programs [18] are both modular and compact not only wrt the states but also wrt the actions. However, in the context of a MAS it is important that actions can have side effects on the states of other agents as well and this behaviour is difficult to model with concurrent programs ([17] contains a detailed comparison). Finally, our choice of using MIS to model MAS is, although motivated by the above reasons, still arbitrary to some extent and our techniques could certainly be used with other formalisms as well.

A major obstacle to model checking real systems is the state explosion problem. As model checking 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. While for small problems this is still feasible, for larger state spaces it soon becomes intractable. We therefore need to eliminate irrelevant states by using appropriate abstraction techniques [4] which guarantee that the property to be verified holds in the original system if it holds for the abstract system. We present such an abstraction technique for MIS. More precisely, we reduce the local state space of each agent in a MIS. We do this by using handcrafted equivalence relations because, clearly, there cannot be a generic automatizable abstraction technique: Model checking ATL for MIS is *EXPTIME*-complete, therefore in the worst case there are instances where no abstraction technique at all is applicable.

While abstraction of reactive systems for temporal properties is a lively research area [2, 3, 7, 19], there are only a few approaches when it comes to MAS and even fewer concerning an abstraction technique for dealing with strategic abilities. One interesting approach by Cohen et al. [6] achieves an abstraction that preserves temporal-epistemic properties. However, the abstraction is based on an interpreted system to model the MAS and therefore limits the modularity of the MAS. Several other abstraction approaches for epistemic properties (cf. [8, 11]) are either not computationally grounded or use an explicit representation of the model.

Another approach by Henzinger et al. [15] shows how to use abstraction

for symbolic model checking of alternating-time μ-calculus formulae over MAS given as alternating transition systems. Their technique 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). Hence, we allow for a much finer control over what information is abstracted away but still preserve soundness of our model checking algorithm.

Note that we will assume the existence of handcrafted equivalence relations, e.g. generated from manual annotations of program code, since any automatic abstraction generation or refinement (as in [14] for two-player games) can only work in typical cases but not in the worst case. That is also the reason why we do not work out how our algorithms can be implemented fully symbolically: In the worst case it will be as bad as a non-symbolical algorithm. We are not trying to neglect the usefulness of either of those techniques but our focus lies on something else: a provable upper bound for the runtime which is exponential in the sum of the sizes of the abstract systems but linear in the size of a succinct representation of the concrete system (see Theorem 6.1). The exponential part of this is not as worse as it sounds because, as argued above, our technique allows for more than one abstraction and therefore each abstraction can be quite small and still much of the relevant information of the whole concrete system can be preserved for the overall model checking process.

Finally, the abstraction for MIS which we present in this paper is motivated by the idea of an IT ecosystem [9], i.e., a system composed of a large number of distributed, decentralized, autonomous, interacting, cooperating, organically grown, heterogeneous, and continually evolving subsystems. In such an ecosystem, which can be seen as a MAS, it is important to verify safety, fairness and liveness properties in order to control the stability of it. A non-trivial demonstrator (mentioned in [9]) describes one instance of such a system by introducing a fictional scenario, namely a smart airport. In that airport there exist many agents doing different things, e.g., carrying your bags, buying flight tickets or exchanging pictures about the travel destination. Among many other things some agents at the airport want to share some information with other agents. Assuming that no direct agent-to-agent connection is possible, the agents have to send the information to some middleman that forwards the message. Obviously this communication protocol raises some questions about safety, fairness and liveness properties. While examining these properties, i.e., model checking the whole system (consisting of many agents and therefore many states), is intractable, model checking a MIS allows us to concentrate on just the agents that have to communicate. Using our abstraction method it is sufficient to model-check a fairly small subset of the original system. We will use this scenario as a running example throughout the paper.

The structure of the paper is as follows: First we present the background of our work. Section 2 recalls the MIS framework and describes our modifications. We extend this section by formulating the communicating agents example as a MIS. In Section 3 we introduce the logic ATL. The main contributions of this paper are in Section 4, Section 5 and Section 6: We design an abstraction technique for MIS, then construct a model checking algorithm based on this technique and conclude with a soundness proof as well as a complexity analysis of the algorithm. Section 7 illustrates the abstraction technique by an example. Finally, Section 8 summarizes the results and discusses future work.

2 Modular Interpreted Systems

We model multi-agent systems in the framework of Modular Interpreted Systems (MIS) [17]. Each agent is described by a set of possible local states, i.e., states it can be in, 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.

Definition 2.1. A *Modular Interpreted System (*MIS*)* is a tuple $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$ where Act is the set of actions all agents can perform. $\mathcal{I}n$ is called interaction alphabet. It describes the interaction between the agent and its environment. Finally, $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_k\}$ is a set of agents where an agent is a tuple $a_i = (St_i, d_i, out_i, in_i, o_i, \Pi_i, \pi_i)$ with

- St_i is the local state space. It is a non-empty set of possible local states for agent a_i.
- $d_i : St_i \to \mathfrak{P}(Act)$ defines for each state in St_i the available actions for agent a_i.
- $out_i : St_i \times Act \to \mathfrak{P}(\mathcal{I}n)$ defines the possible influences (one is then chosen non-deterministically) of agent a_i's action (executed in a certain local state) on its environment.[1] Intuitively, this describes the external effect of an action which agent a_i is executing.
- $in_i : St_i \times \mathcal{I}n^{k-1} \to \mathfrak{P}(\mathcal{I}n)$ defines the possible influences (one is then chosen non-deterministically) of its environment on this agent.[1] It

[1] This is different from the original MIS definition in so far as we have a set of possible influences and the authors had one deterministic influence symbol; it is changed to cope with possible ambiguities when doing abstraction later.

maps the external effects of the actions of all other agents to the influence these actions might have on the agent in a particular state.

- $o_i : St_i \times Act \times \mathcal{I}n \to \mathfrak{P}(St_i)$ is a local (non-deterministic) transition function.[2].
- Π_i are the local propositions, where Π_i and Π_j are disjoint when $i \neq j$.
- $\pi_i : St_i \to \mathfrak{P}(\Pi_i)$ is a valuation (local labeling function) of these propositions.

The global state space is defined as $St := St_1 \times \cdots \times St_k$.

Example 2.2. We consider a system with several autonomous agents which can gather information about their environment and share that information between each other if they are in communication range. We consider groups of them working together as teams.

Our example consists of six agents $\{a_1, a_2, a_3, a_4, b_1, b_2\}$ partitioned in two teams $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2\}$ and where the agents' locations are such that each agent a_i can reach each agent b_j but no two agents from the same team can reach each other (see Figure 1). Agents of team A can send a message (if they already know it) to b_1 or b_2 or choose to do nothing. Agents of team B, however, are not allowed to send a message back to its sender agent. Once an agent a_i sent a message to an agent b_j the agent b_j is not allowed to send it to a_i in any future round. Additionally, if an agent b_j has received a message from a_i then it has to send it to some agent a_k in the following round (unless this contradicts the former rule) and if possible k has to be greater than i.

Now agent a_1 has learned something and wants to communicate its newly gathered knowledge to its team member a_4. The difficulty is that the message has to pass through an agent of the other team. But we will see that it is still possible for team A to ensure that a_4 will know the message eventually.

Formally we have the following MIS

$$\begin{aligned} S := (\mathbb{A}\mathrm{gt} = \{a_1, a_2, a_3, a_4, b_1, b_2\}, Act = \{\mathrm{send_x} \mid \mathrm{x} \in \mathbb{A}\mathrm{gt}\} \cup \{\mathrm{noop}\}, \\ \mathcal{I}n = \{\mathbf{nothing}, \mathbf{m_{a_1}}, \mathbf{m_{a_2}}, \mathbf{m_{a_3}}, \mathbf{m_{a_4}}\} \\ \cup \mathfrak{P}(\{\mathbf{m_{a_i b_j}} \mid i \in \{1, \ldots, 4\}, j \in \{1, 2\}\})) \end{aligned}$$

with $a_i := (St_{a_i}, d_{a_i}, out_{a_i}, in_{a_i}, o_{a_i}, \Pi_{a_i}, \pi_{a_i})$ where

- $St_{a_i} = \{k(nown), u(nknown)\}$
- $\Pi_{a_i} = \{\mathsf{known}_{\mathsf{a_i}}, \mathsf{unknown}_{\mathsf{a_i}}\}$
- $\pi_{a_i} : \quad k \quad \mapsto \quad \{\mathsf{known}_{\mathsf{a_i}}\}, \quad u \mapsto \{\mathsf{unknown}_{\mathsf{a_i}}\}$

[2]Non-deterministic as opposed to in the original definition; it inherits the non-determinism from in_i and adds additional non-determinism to cope with abstraction of states later.

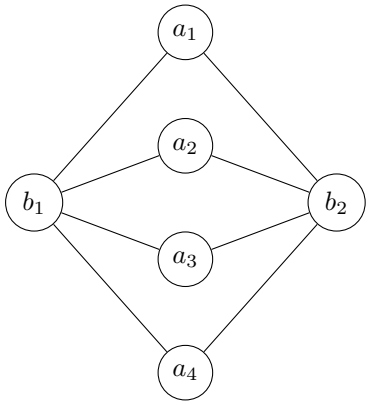

Figure 1: Communication graph of the six agents

- $d_{a_i}: \quad k \mapsto \{\mathrm{send}_{\mathrm{b}_1}, \mathrm{send}_{\mathrm{b}_2}, \mathrm{noop}\}, \quad u \mapsto \{\mathrm{noop}\}$
- $out_{a_i}: \quad \begin{array}{lll} (k, \mathrm{send}_{\mathrm{b_j}}) & \mapsto & \{\{\mathbf{m_{a_i b_j}}\}\} \\ (k, \mathrm{noop}) & \mapsto & \{\mathbf{nothing}\} \\ (u, \mathrm{noop}) & \mapsto & \{\mathbf{nothing}\} \end{array}$
  for all $j \in \{1, 2\}$,
- $in_{a_i}: \quad (s, \gamma_1, \ldots, \gamma_5) \mapsto \begin{cases} \{\mathbf{m_{a_i}}\} & \text{if } \mathbf{m_{a_i}} \in \{\gamma_1, \ldots, \gamma_5\} \\ \{\mathbf{nothing}\} & \text{else} \end{cases}$
  for all $s \in St_{a_i}$, $\gamma_1, \ldots, \gamma_5 \in \mathcal{I}n$,
- $o_{a_i}: \quad \begin{array}{lll} (k, \alpha, \gamma) & \mapsto & \{k\} \\ (u, \alpha, \mathbf{nothing}) & \mapsto & \{u\} \\ (u, \alpha, \mathbf{m_{a_i}}) & \mapsto & \{k\} \end{array}$
  for all $\gamma \in \mathcal{I}n$ and $\alpha \in Act$.

For the agents b_j we have $b_j := (St_{b_j}, d_{b_j}, out_{b_j}, in_{b_j}, o_{b_j}, \Pi_{b_j}, \pi_{b_j})$ where

- $St_{b_j} = \mathfrak{P}(\{r_1, \ldots, r_4\}) \times \mathfrak{P}(\{n_1, \ldots, n_4\})$,
- $\Pi_{b_j} = \{\mathsf{known}_{\mathsf{b_j}}, \mathsf{unknown}_{\mathsf{b_j}}\}$,
- $\pi_{b_j}: (R, N) \mapsto \begin{cases} \{\mathsf{known}_{\mathsf{b_j}}\} & \text{if } R \neq \emptyset \\ \{\mathsf{unknown}_{\mathsf{b_j}}\} & \text{else} \end{cases}$,
- $d_{b_j}: (R, N) \mapsto \begin{cases} \{\mathrm{noop}\} & \text{if } R = \emptyset \\ \left\{ \mathrm{send}_{\mathrm{a_i}} \,\middle|\, \begin{array}{l} r_i \notin R \text{ and there is no } k \geq i\text{:} \\ n_k \in N \text{ and } \exists \ell > k\text{: } r_\ell \notin R \end{array} \right\} & \\ \quad \cup \begin{cases} \{\mathrm{noop}\} & \text{if } N = \emptyset \text{ or } R = \{r_1, \ldots, r_4\} \\ \emptyset & \text{else} \end{cases} & \text{else} \end{cases}$,

- $out_{b_j}: \begin{array}{lcl} ((R,N),\mathrm{send}_{\mathrm{a_i}}) & \mapsto & \{\mathbf{m_{a_i}}\} \\ ((R,N),\mathrm{noop}) & \mapsto & \{\mathbf{nothing}\} \end{array}$
  for all $i \in \{1,\ldots,4\}$,

- $in_{b_j}: ((R,N),\gamma_1,\ldots,\gamma_5) \mapsto \begin{cases} \{\{\mathbf{m_{a_i b_j}} \mid \{\mathbf{m_{a_i b_j}}\} \in \{\gamma_1,\ldots,\gamma_5\}\}\} \\ \quad \text{if there is } i \in \{1,\ldots,4\} \text{ with } \{\mathbf{m_{a_i b_j}}\} \in \{\gamma_1,\ldots,\gamma_5\} \\ \{\mathbf{nothing}\} \quad \text{else} \end{cases}$
  for all $\gamma_1,\ldots,\gamma_5 \in \mathcal{I}n$,

- $o_{b_j}: \begin{array}{lcl} ((R,N),\alpha,M) & \mapsto & (R \cup \{r_i \mid \mathbf{m_{a_i b_j}} \in M\}, \{r_i \mid \mathbf{m_{a_i b_j}} \in M\}) \\ ((R,N),\alpha,\mathbf{nothing}) & \mapsto & (R,\emptyset) \end{array}$
  for all $\alpha \in Act$, $\emptyset \neq M \subseteq \{\mathbf{m_{a_1 b_j}},\ldots,\mathbf{m_{a_4 b_j}}\}$,

for all $(R,N) \in St_{b_j}$. R stands for "**r**eceived (now or some time ago)" while N means "received **n**ow".

Figure 2 shows the agent a_1. Each arrow is denoted by an action and an incoming interaction symbol. Outgoing symbols and propositions are omitted for ease of representation. Nevertheless state u should be labeled with $\mathsf{unknown}_{\mathsf{a_1}}$ and k with $\mathsf{known}_{\mathsf{a_1}}$. The agents a_i are fairly simply structured, they consist of two states representing whether the agent knows the message (k) or it does not know it (u). In the former case the agent can send the message to one of the opponents or just do nothing. In the latter case it has to wait for some agent of team B sending the message to it.

The structure of the agent b_j however is more complex since it consists of 256 states. Every state is labeled with $\mathsf{known}_{\mathsf{b_j}}$ if the state name contains at least one r_i, i.e., the agent received some time ago the message from agent a_i. Consequently, states that do not have any r_i are marked as $\mathsf{unknown}_{\mathsf{b_j}}$. Intuitively, while the agent is waiting for a message it does nothing. When it receives a message, i.e., the state contains a n_i it has to send the message to one of the opponents with a higher number then i and with the condition that this agent did not send it to b_j before. If the state contains all r_1 to r_4 the agent does nothing.

We will come back to this concrete example when presenting the logic and the model checking algorithm.

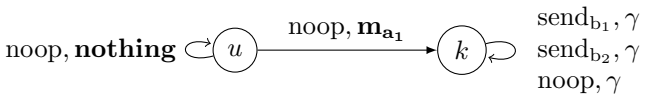

Figure 2: Graph of agent a_1

3 Specification Logic ATL

After having outlined the framework with which we model our MAS, we now have to specify a logic to talk about strategic properties of such a system. We recall the syntax of ATL, define some abbreviations and sketch the semantics of ATL for MIS.

Definition 3.1. Alternating-time temporal Logic (ATL) [1] is a logic that enables reasoning about temporal and strategic abilities of multi-agent systems. The syntax of plain ATL is inductively defined by the following BNF (with $A \subseteq \mathbb{A}\mathrm{gt}$)

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\langle A\rangle\rangle \mathbf{X}\varphi \mid \langle\langle A\rangle\rangle \mathbf{G}\varphi \mid \langle\langle A\rangle\rangle \varphi \mathbf{U} \varphi.$$

Informally, $\langle\langle A\rangle\rangle \mathbf{X}\varphi$ means that agents A have a collective strategy to enforce that in the next step φ holds. The operator $\mathbf{X}$ is read "in the ne**x**t state", the symbol $\mathbf{G}$ means "**g**lobally" and $\mathbf{U}$ "**u**ntil". Other Boolean operators are defined by macros in the usual way.

For the model checking algorithm presented in Section 5 we need to split a formula into subformulae.

Definition 3.2. For an ATL formula φ we write $\mathrm{qsf}(\varphi)$ for the multiset of all subformulae of φ which start with a quantifier $\langle\langle A\rangle\rangle$.

Note that identical formulae occurring in two different places inside φ occur twice in $\mathrm{qsf}(\varphi)$ and if φ itself begins with a quantifier it is in the multiset as well.

Finally, we need the following notions.

Definition 3.3. For a formula $\varphi = \langle\langle A\rangle\rangle\psi$ we write $[\![\varphi]\!]$ for the set A of agents. For an arbitrary formula φ and an arbitrary $\psi \in \mathrm{qsf}(\varphi)$ let $\varphi(\psi, w)$ denote the formula resulting from φ by replacing ψ with the new proposition w.

Example 3.4. Considering the communicating agents example (cf. Example 2.2) we can ask the following question: Is it possible for team A to ensure that a_4 will know the message eventually? Written in ATL this corresponds to the question whether

$$S, \mathrm{q} \models \langle\langle A\rangle\rangle(\top \mathbf{U} \mathsf{known}_{\mathsf{a_4}})$$

with $A = \{a_1, a_2, a_3, a_4\}$ and q is the global state where a_1 is in state k, all other agents from team A are in state u and the agents from team B are in state $(\emptyset, \emptyset)$, i.e. where only a_1 knows the message.

3.1 Semantics of ATL for MIS

Modular Interpreted Systems can be easily transformed into *concurrent game structures* (CGS, cf. [1]) as shown for deterministic CGS in [17]. The notion of a CGS is a very universal formalism to model MAS but it comes at the cost of not being modular, i.e., CGS have an unstructured global state space. Also, just as MIS, they have for every agent i a function $d_i : St \to Act$ expressing which actions are available to agent i in a certain global state. The global transition function of a CGS takes as input a global state and for every agent a permissible action and outputs a set of possible successor states (one is then chosen non-deterministically), i.e., the influences of an agent on its environment are implicitly given and there is no way to measure or limit that influence in the framework of CGS.

Definition 3.5. For a MIS $S = (\{a_1, \ldots, a_k\}, Act, \mathcal{I}n)$ with $a_i = (St_i, d_i, out_i, in_i, o_i, \Pi_i, \pi_i)$ $(1 \leq i \leq k)$ the corresponding (non-deterministic) concurrent game structure $ncgs(S) = (\mathbb{A}\mathrm{gt}', St', \Pi', \pi', Act', d', o')$ is defined as follows:

- $\mathbb{A}\mathrm{gt}' := \{1, \ldots, k\}$
- $St' := \prod_{i=1}^{k} St_i$
- $\Pi' := \Pi_1 \cup \cdots \cup \Pi_k$
- $\pi'(p) := \{(q_1, \ldots, q_i, \ldots, q_k) \mid q_i \in \pi_i(p)\}$
- $Act' := Act$
- $d'_i((q_1, \ldots, q_k)) := d_i(q_i)$
- $\delta'((q_1, \ldots, q_k), \alpha_1, \ldots, \alpha_k) :=$
  $\{(q'_1, \ldots, q'_k) \mid \forall 1 \leq i \leq k : q'_i \in o_i(q_i, \alpha_i, \gamma_i)$ for a $\gamma_i \in in_i(q_i, \gamma_1, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_k)$ for some $\gamma_1, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_k$ with $\gamma_j \in out_j(q_j, \alpha_j)\}$

Note that our concurrent game structures extend the definition from [1] in two ways. Firstly we use labeled actions instead of plain numbers. Secondly we allow the transition relation to be non-deterministic. The semantics of a formula $\langle\langle A\rangle\rangle\varphi$ over a non-deterministic CGS is defined with the non-determinism working against the agents in A. The rationale behind this is that $\langle\langle A\rangle\rangle\varphi$ means "the agents A have a combined strategy which enforces φ". Now, to enforce φ this strategy needs to ensure that in each of the possible runs of the system – determined by the other agents' choices and the non-deterministic branching – the formula φ holds.

Model checking ATL for deterministic MIS is *EXPTIME*-complete as stated in [16] and for deterministic CGS it is *PTIME*-complete as stated in [1]. These results still hold for the non-deterministic versions of the structures. This is

because in the model checking algorithm from [1, Chapter 4.1] introducing non-determinism only changes the function $\mathrm{Pre}(A, \rho)$ (which for a given set ρ of system states and a given set A of agents outputs the set of system states from which the agents A can enforce that the next state in any run will lie in ρ). And computing Pre does not get more difficult with non-determinism because even in the case of deterministic systems it is already necessary to take into account all transitions in order to compute Pre.

Having defined MIS and ATL we can now present our new abstraction technique.

4 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. Under this assumption it makes sense to reduce the state space by removing irrelevant states and/or by combining them. Due to the modularity of MIS, we can in a first step easily remove the obviously non-relevant parts of the global state space by removing particular agents while keeping the others. Secondly, we reduce the state space of each agent by abstraction. As in [4] and [6] 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 just as the concrete one. The set of available actions in an abstract state is decreased so that it only contains actions available in every one of the equivalent concrete states. Finally we show how to handle the interaction with an agents' environment. We start by introducing the definition of an abstraction relation.

Definition 4.1. An *abstraction relation* for a MIS is a product $\equiv = \equiv_1 \times \cdots \times \equiv_k$ where each $\equiv_i \subseteq St_i \times St_i$ is an equivalence relation for the states St_i of agent a_i.

For $q \in St_i$, we write $[q]_{\equiv_i}$ for the equivalence class of the local state q with respect to $\equiv_i$. And for $q \in St = St_1 \times \cdots \times St_k$, we write $[q]_{\equiv}$ for the equivalence class of the global state q.

An abstraction relation as in Definition 4.1 defines for each agent of the MIS, which local states are equivalent and therefore can be condensed to one abstract state. Note that this definition does not say anything about how to define the equivalence, because this depends on the concrete system that is model checked. Therefore these relations have to be handcrafted when modeling a system.

Using the definition of a MIS and Definition 4.1 we can now specify the abstraction of a system:

Definition 4.2. For a MIS $S = (\mathbb{A}\mathrm{gt}, Act, \mathcal{I}n)$, an abstraction relation $\equiv$ for S and a set of *favored* agents $A \subseteq \mathbb{A}\mathrm{gt}$ we define the *abstraction* of S with respect to $\equiv$ and A as the MIS

$$S^A_{\equiv} := (\mathbb{A}\mathrm{gt}', Act, \mathcal{I}n)$$

where $\mathbb{A}\mathrm{gt}' := \{a'_1, \ldots, a'_k\}$ and $a'_i = (St'_i, d'_i, out'_i, in'_i, o'_i, \Pi'_i, \pi'_i)$ with

i) $St'_i := \{[q]_{\equiv_i} \mid q \in St_i\}$

ii) $d'_i([q]_{\equiv_i}) := \begin{cases} \bigcap_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \in A \\ \bigcup_{q' \in [q]_{\equiv_i}} d_i(q') & \text{for } a_i \notin A \end{cases}$

iii) $out'_i([q]_{\equiv_i}, \alpha) := \bigcup_{q' \in [q]_{\equiv_i}} out_i(q', \alpha)$

iv) $in'_i([q]_{\equiv_i}, \gamma_1, \ldots, \gamma_{k-1}) := \bigcup_{q' \in [q]_{\equiv_i}} in_i(q', \gamma_1, \ldots, \gamma_{k-1})$

v) $o'_i([q]_{\equiv_i}, \alpha, \gamma) := \bigcup_{q' \in [q]_{\equiv_i}} \{[q'']_{\equiv_i} \mid q'' \in o_i(q', \alpha, \gamma)\}$

vi) $\Pi'_i := \Pi_i \cap \{p_i \mid \forall q \in \pi_i(p_i) : \forall q' \in [q]_{\equiv_i} : q' \in \pi_i(p_i)\}$

vii) $\pi'_i(p_i) := \{[q]_{\equiv_i} \mid q \in \pi_i(p_i)\}$

for all $q \in St_i$, $\alpha \in Act$, $\gamma, \gamma_1, \ldots, \gamma_k \in \mathcal{I}n$ and $p_i \in \Pi'_i$.

Note that there might be $i \in \{1, \ldots, k\}$ and $q \in St_i$ such that $d'_i([q]_{\equiv_i}) = \emptyset$. As this would paralyse the system we will from now on assume that the abstraction relation is chosen in such a way that this does not happen.

Formula i) defines a partition of the local state space by using the handcrafted equivalence relation of this agent. We reduce all equivalent states to just one. Function ii) then computes for this element the available actions by giving agents in A fewer choices and the opponents more choices than before. Due to this construction if a property $\langle\langle A \rangle\rangle \varphi$ (with φ propositional) holds in the abstract system it also holds in the concrete one, since we restricted the actions of the protagonists and extended the set of actions of the antagonists. The possible influences of these actions concerning the environment are calculated by the resulting function iii). It takes for the action α the union of all resulting influence symbols of all states in the equivalence class, i.e., collecting all influence symbols that are an outcome of executing action α in each state q' of the equivalence class $[q]_{\equiv_i}$. Taking the union is motivated by the fact that executing the same action in equivalent local states results in an equivalent influence on the environment. iv) is defined the same way: We just use the union of in_i for each state in the equivalence class. Moreover, the out_i- and in_i-functions are of the same type for

the protagonists as well as the antagonists because they do not introduce deliberate choices by the agents but instead introduce nondeterminism which will – as we will see in Section 5 – always work against the formula which is to be verified. The local transition function v) has to be modified to cope with equivalence classes: It gets as input an abstract state, an action and one (nondeterministically chosen) influence symbol. The output is the set of all equivalence classes that are successors. To determine this we unfold both equivalence classes and check whether there is a connection between a concrete state of the first equivalence class to another concrete state of the second equivalence class.

Finally, we have to define how to label the abstract states (cf. vi) and vii)). We do this by assigning a proposition to an abstract state if all concrete states in the equivalence class are labeled with that proposition. If a proposition only holds in some states of the class we remove it from set of propositions. This ensures that if a proposition is true in an abstract state it is also true in all concrete ones.

In the next section we describe how to evaluate whether a formula holds in a system.

5 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 $\psi \in \mathrm{qsf}(\varphi)$ an abstraction relation $\equiv_\psi$. The algorithm 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.

This behaviour is due to the way model checking is done here: Several abstractions of S (generated out of the abstraction relations $\equiv_\psi$) are used each to model check a part of φ. And as usual with handcrafted abstractions there can be false negatives. The important point is that there are no false positives, i.e. if the abstractions fulfill φ then so does the concrete system.

Before we can present the algorithm we need the technical notion of a pseudo-MIS which will be used in it.

Definition 5.1. A *pseudo*-MIS is a MIS together with a set Π of *global propositions* (which is disjoint to each set of local propositions) and a global labeling function $\pi : St \to \mathfrak{P}(\Pi)$. Note that every MIS can be viewed as a pseudo-MIS with $\Pi = \emptyset$.

The algorithm now works as follows. Details about efficiently implementing some of the steps are given in the proof of Theorem 6.1.

Algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$:
Let $\varphi = \lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- λ is a monotone Boolean formula, i.e. λ is composed of conjunctions and disjunctions only,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, i.e. each θ_i is of the form $\langle\langle B\rangle\rangle\theta_i'$ or $\neg\langle\langle B\rangle\rangle\theta_i'$ (in the latter case we will still write $\equiv_{\theta_i}$ and $[\![\theta_i]\!]$ instead of $\equiv_{\langle\langle B\rangle\rangle\theta_i'}$ and $[\![\langle\langle B\rangle\rangle\theta_i']\!]$), and
- $\ell_1, \ldots, \ell_m$ are literals, i.e. atomic propositions or negations of atomic propositions.

1) For all $i \in \{1, \ldots, n\}$ do:
   i) Set $W_i := \mathit{label}(\theta_i, \equiv_{\theta_i})$.
   ii) Set $S := S(w_i, W_i)$, i.e. S is from now on viewed as a pseudo-MIS, a new global proposition w_i is introduced in S and it is labeled exactly in the states in W_i.
2) If $S, s \models \lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ for all $s \in \mathit{init}$ then return true. Otherwise return unknown.
   Note that for this step the algorithm only has to locally check the labeling of the states $s \in \mathit{init}$ as $\lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)$ is an entirely propositional formula.

Algorithm *label*$(\psi, \equiv)$:
Let $\psi = \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y}\lambda(\theta_1, \ldots, \theta_n, \ell_1, \ldots, \ell_m)$ where

- $\neg^{\psi}$ is $\neg$ if ψ begins with a negation and it is the empty string elsewise,
- $\mathbf{Y} \in \{\mathbf{X}, \mathbf{G}, \mathbf{U}\}$,
- λ is a monotone Boolean formula,
- $\theta_1, \ldots, \theta_n$ are arbitrary ATL formulae each beginning with a quantifier or a negation directly followed by a quantifier, and
- $\ell_1, \ldots, \ell_m$ are literals.

1) Construct the abstraction

$$S' := \begin{cases} S_{\equiv}^{[\![\psi]\!]} & \text{if } \psi \text{ does not begin with a negation} \\ S_{\equiv}^{\mathbb{A}\mathrm{gt}\setminus[\![\psi]\!]} & \text{if } \psi \text{ does begin with a negation} \end{cases}$$

   We will view S' as a pseudo-MIS in the following steps.
2) For all $i \in \{1, \ldots, n\}$ do:

i) Set $W_i := \{[q]_{\equiv} \mid \forall q' \in [q]_{\equiv} : q' \in \mathit{label}(\theta_i, \equiv_{\theta_i})\}$.

ii) Set $S' := S'(w_i, W_i)$.

3) Compute the set W' of global states of S' (note that these are global states of the system abstracted with $\equiv$) satisfying ψ, i.e. $W' :=$

$$\{[q]_{\equiv} \mid S', [q]_{\equiv} \models \neg^{\psi}\langle\langle A\rangle\rangle \mathbf{Y}\lambda(w_1, \ldots, w_n, \ell_1, \ldots, \ell_m)\},$$

by translating S' to a non-deterministic CGS and then using the ATL model checking algorithm from [1, Chapter 4.1]. As already pointed out in Section 3.1 their algorithm is only given for deterministic CGS but can be easily adapted to also handle non-deterministic systems.

There is, however, a caveat here. Because additional non-determinism might be introduced by abstracting the system we have to make sure that the non-determinism works "against the formula" because we want to avoid false positive outputs of our algorithm. This is the reason why we have to interpret the non-determinism as working *for* the agents in A if $\neg^{\psi} = \neg$ and working *against* them otherwise. If we always had it working against them (which seems natural as argued in Section 3.1) then in the former case it could happen that the algorithm comes to the conclusion that $S', [q]_{\equiv} \models \psi$ although $S, q \not\models \psi$ – a false positive. The reason for this would be non-determinism present in S' and absent in S that would presumably prevent agents A to have a winning strategy in S' although they do have one in S.

4) Return $W := \{q \in St \mid [q]_{\equiv} \in W'\}$.

6 Complexity and Soundness of the Algorithm

The following theorem shows that our model checking algorithm runs in time linear in the size of a succinct representation of the concrete system as well as linear in the length of the formula and exponential in the sum of the sizes of the abstract systems. Now, since there is a special abstraction for each modality, the abstractions should be very small and therefore this should be a huge improvement over the *EXPTIME*-completeness of model checking MIS without abstractions.

Theorem 6.1. *Algorithm* modelcheck$(S, init, \varphi, (\equiv_{\psi})_{\psi \in \mathrm{qsf}(\varphi)})$ *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* $2^{\Theta(|S|)}$.

Proof. The crucial implementation detail is that it is not possible to explicitly enumerate the set returned in step 4 of *label*() because the set may be as large as the global state space St of S. Instead, both *label*() and *modelcheck*() have to save and pass on computed sets of global states in a symbolic way, i.e. by refering to the modular structure of S and to the abstraction relation's equivalence classes. This technique is needed in step 1 of *modelcheck*() and in steps 2i and 4 of *label*().

Also, in step 2i of *label*() it is not possible to check the condition for each $[q]_{\equiv} \in St'$ by enumerating through all $q' \in [q]_{\equiv}$ because a single equivalence class may already be as large as the global state space. Hence, the algorithm has to construct the equivalence relation $\equiv' := \equiv \cap \equiv_{\theta_i}$ (which is a refinement of both $\equiv$ and $\equiv_{\theta_i}$) and compute W_i as the set $\{[q]_{\equiv} \mid \forall [q']_{\equiv'} \subseteq [q]_{\equiv} : [q']_{\equiv'} \subseteq \mathit{label}(\theta_i, \equiv_{\theta_i})\}$. This can be done in time $O(|S^{[\![\psi]\!]}_{\equiv}| \cdot |S^{[\![\theta_i]\!]}_{\equiv_{\theta_i}}|)$.

Step 3 of *label*() runs in time $2^{O\left(|S^{[\![\psi]\!]}_{\equiv}|\right)} \cdot O(|\psi|)$ because the translation to a CGS may involve an exponential blow-up in the system size. All other steps are easy to implement – when keeping in mind the symbolic handling of state sets.

Finally, *label*() is executed at most $|\varphi|$ times. Altogether this gives the claimed upper bound on the runtime. □

The following theorem shows that our algorithm is sound. It is, however, not complete because, as usual for abstraction techniques, the capability of the algorithm to show the truth of a formula depends on choosing a suitable abstraction. It should, anyhow, be possible to find good abstractions since it is possible to define a specific abstraction for each strategic operator. Of course that problem could be overcome by an automatic abstraction refinement technique but this, on the other hand, would make a provable upper bound on the runtime in the form of Theorem 6.1 impossible.

Theorem 6.2. *Algorithm modelcheck is sound, i.e. if* modelcheck$(S, init, \varphi, (\equiv_\psi)_{\psi \in \mathrm{qsf}(\varphi)})$ *outputs* true *then* $S, q \models \varphi$ *for all* $q \in init$.

Proof. (Sketch) First note that if we skipped step 1 of the *label*() algorithm and simply ran the algorithm without constructing any abstractions we would exactly run the bottom-up, subformula labeling, model checking algorithm from [1, Chapter 4.1].

Hence we only have to argue why the abstractions do not lead the algorithm to produce more positive answers than without them. The crucial observation is that for each modality $\langle\langle A \rangle\rangle \mathbf{Y}$ the aspects which are of an existentially quantifying nature, i.e. the actions available to agents A, can only be restricted by an abstraction but never extended and for the aspects of universally quantifying nature, i.e. the actions available to agents $\mathbb{A}\mathrm{gt} \setminus A$ as well as the non-deterministic branching of the system, it is the other way around.

Thus it is ensured that if a formula $\langle\langle A\rangle\rangle \mathbf{Y}\varphi$ is true in an abstraction it is also true in the original system.

Furthermore, for formulae of the form $\neg\langle\langle A\rangle\rangle \mathbf{Y}\varphi$ the abstraction is constructed the other way around, i.e. extended choices for A and restricted choices for $\mathbb{A}\mathrm{gt} \setminus A$, to ensure that if the agents A do not have a winning strategy in the abstraction then neither do they have one in the original system.

The non-determinism, however, is extended even in this case. As already discussed in step 3 of the algorithm we therefore have to change the meaning of the non-determinism to be of existential rather than of universal nature. The sacrifice we have to make is that if the original system is already non-deterministic and this non-determinism ensures some property $\neg\langle\langle A\rangle\rangle \mathbf{Y}\varphi$ then the algorithm will return unknown. □

7 Communicating Agents Example

Consider the example of the six agents again (Example 2.2 and Example 3.4). We will now apply the model checking algorithm to the example using the formula

$$S, \mathrm{q} \models \langle\langle A\rangle\rangle(\top \mathbf{U} \mathsf{known}_{\mathsf{a}_4})$$

with $A = \{a_1, a_2, a_3, a_4\}$ and q is the global state in which only a_1 knows the message (cf. Example 3.4). The formula φ describes the following question: "Is it possible for team A to always ensure that a_4 will know the message eventually?" The algorithm takes the formula φ and constructs for all quantifier subformulae an abstract system by using the specific abstraction relation for that quantifier. The multiset $\mathrm{qsf}(\varphi)$ of quantified subformulae consists just of the formula φ. Therefore, we have to define only one abstraction for φ.

Before we give the abstraction relation we note that b_2 is not necessary for the property we want to verify and therefore we can temporarily delete it from the system. As abstraction for φ we do not abstract the agents a_i at all and for agent b_1 we use the equivalence relation given by the following partition of its local state space:

$$\begin{array}{lcll} S_i & := & \{(R,N) \mid \emptyset \neq R \subseteq \{r_1,\ldots,r_i\}, n_i \in N\} \setminus \bigcup_{j=1}^{i-1} S_j & \text{for } i = 1,\ldots,3 \\ S_{\mathrm{rest}} & := & \{(R,N) \mid (R,N) \notin S_1 \cup \cdots \cup S_3\} & \end{array}$$

Now, the agents a_i remain unchanged and the abstracted agent b_1 looks like the following:

$$b_1' = (St_{b_1}', d_{b_1}', out_{b_1}', in_{b_1}, o_{b_1}', \Pi_{b_1}, \pi_{b_1}')$$

where

- $St_{b_1}' \quad = \quad \{S_1, S_2, S_3, S_{\mathrm{rest}}\}$

- π'_{b_1} : $S_i \mapsto \{\mathsf{known}_{\mathsf{b}_1}\}$ for $i = 1, \ldots, 3$
  $S_{\text{rest}} \mapsto \{\mathsf{known}_{\mathsf{b}_1}, \mathsf{unknown}_{\mathsf{b}_1}\}$

- $d'_{b_1}(S_i) = \{\text{send}_{\text{x}} \mid \text{x} \in \{a_{i+1}, \ldots, a_4\}\}$ for $i = 1, \ldots, 3$
  $d'_{b_1}(S_{\text{rest}}) = \{\text{send}_{\text{x}} \mid \text{x} \in \{a_1, \ldots, a_4\}\} \cup \{\text{noop}\}$

- out'_{b_1} : $(S_{\text{rest}}, \text{noop}) \mapsto \{\mathbf{nothing}\}$
  $(s, \text{send}_{\text{a}_1}) \mapsto \{\mathbf{m_{b_1 a_1}}\}$
  $(s, \text{send}_{\text{a}_2}) \mapsto \{\mathbf{m_{b_1 a_2}}\}$
  $(s, \text{send}_{\text{a}_3}) \mapsto \{\mathbf{m_{b_1 a_3}}\}$
  $(s, \text{send}_{\text{a}_4}) \mapsto \{\mathbf{m_{b_1 a_4}}\}$
  for all $s \in St'_{b_1}$,

- o'_{b_1} : $(S_{\text{rest}}, \alpha, \mathbf{nothing}) \mapsto \{S_{\text{rest}}\}$
  $(S_{\text{rest}}, \alpha, \{\mathbf{m_{a_j b_1}}\}) \mapsto \{S_{\text{rest}}, S_j\}$
  $(S_i, \text{send}_{\text{a}_j}, \mathbf{nothing}) \mapsto \{S_{\text{rest}}\}$
  $(S_i, \text{send}_{\text{a}_j}, \{\mathbf{m_{a_{j'} b_1}}\}) \mapsto \{S_{j'}\}$
  $(S_i, \text{send}_{\text{a}_j}, \{\mathbf{m_{a_4 b_1}}\}) \mapsto \{S_{\text{rest}}\}$
  for all $\alpha \in Act$, $\gamma \in \mathcal{I}n$, $i, j' \in \{1, 2, 3\}$ and $j \in \{1, 2, 3, 4\}$.

Now, everything is specified so that the algorithm *modelcheck*$(S, init, \varphi, (\equiv_\psi)_{\psi \in \text{qsf}(\varphi)})$ can be started. S is the MIS mentioned in Section 2, $init = \{q\}$, φ as above and $(\equiv_\psi)_{\psi \in \text{qsf}(\varphi)}$ contains all abstraction relations for the quantified subformulae. Since the formula φ only consists of one quantifier, by invoking the algorithm *modelcheck*() we get the quantifier subformulae $\theta_1 := \varphi$. The algorithm takes then φ and computes W_1 by executing the labeling algorithm *label*$(\varphi, \equiv_\varphi)$. Now, we construct the pseudo-MIS $S'_\varphi := S^{[\![\varphi]\!]}_{\equiv_\varphi}$ for the favored agents a_1, a_2, a_3, a_4. Step 2i) of the labeling algorithm is skipped since there is no further quantified subformula for φ. This is the moment when the recursion stops and we start to label the states in a bottom-up order. $W_1 := \{[q]_{\equiv_\varphi} \mid S'_\varphi, [q]_{\equiv_\varphi} \models \varphi\}$ is computed by creating the non-deterministic CGS and apply the model checking algorithm for ATL.

Now we are almost finished. In the *modelcheck*() algorithm we set $S' := S'(w_1, W_1)$. The last step is to evaluate whether $S, s \models \varphi$ holds and we therefore answer with true.

8 Conclusion and Future Work

While in the MAS community model checking agent systems already has attracted some attention there has not been much work on abstraction techniques for reducing the state space. In this paper, we presented a technique to cope with the state explosion problem which opens the way to reduce

the state space of a MAS so that model checking might get tractable. Clearly, there cannot be a generic automatizable abstraction technique: Model checking ATL for MIS is *EXPTIME*-complete, therefore in the worst case, there are instances where no abstraction technique at all is applicable.

Consequently we focused on handcrafted 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 abstraction always leads to losing completeness. However, abstraction still has its benefits because without reducing the state space many problems could not be model checked at all. Defining different abstraction relations for each quantifier allows to shrink the state space as needed for each subformula. Usually a MAS consists of more than two teams and the agents are more complex than in our example which increases our speedup factor significantly. In fact, we believe that most real problems carry with them a rich structure which allows the abstraction technique to be successfully applied, especially when using the possibility to use more than one abstraction for a single formula.

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 replacing or 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.

The need to have a compact, modular and grounded representation was motivated by the idea of an IT ecosystem, i.e., a system composed of a large number of distributed, decentralized, autonomous, interacting, cooperating, organically grown, heterogeneous, and continually evolving subsystems. An example for such a system is a smart city that contains agents for cars, traffic lights, cameras, etc. In such an IT ecosystem, new agents are introduced, other agents are removed and others again are modified. If we nevertheless want to ensure some safety, liveness or fairness properties we need a framework that on the one hand enables theoretical analysis and on the other hand supports modularity.

An IT ecosystem in general is the topic of a large research project consisting of 17 professors and 33 scientists in total collecting knowledge in different research areas: multi-agent systems, organic computing, ambient intelligence, software engineering and embedded systems. Together we try to solve the contradiction of having a continually evolving and highly heterogeneous system on the one side and still controlling this system by ensuring some properties on the other side.

We are currently implementing our abstraction technique in a first prototype and will use it for a concrete, non-trivial demonstrator scenario [9]. The application will run on a smartphone and will send properties to be checked to a server that will then model check it. Users will get feedback if the for-

mula holds or if it is unknown and can then decide whether they want to take part in the IT ecosystem. For this implementation it will, of course, be very useful to develop some heuristics and automatic refinement methods to generate abstractions.

For the future, we plan to put some effort in developing parallel model checking methods for this system and using a logic that facilitates the use of probabilities.

9 Acknowledgments

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.

References

[1] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM 49(5), 672–713 (2002)

[2] Ball, T., Rajamani, S.: Boolean programs: A model and process for software analysis. Tech. Rep. 2010-14 (2000)

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

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

[5] Clarke, E., Grumberg, O., Peled, D.: Model checking. Springer (1999)

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

[7] Das, S., Dill, D.: Successive approximation of abstract transition relations. In: Logic in Computer Science, 2001. Proceedings. 16th Annual IEEE Symposium on. pp. 51–58. IEEE (2002)

[8] Dechesne, F., Orzan, S., Wang, Y.: Refinement of kripke models for dynamics. In: Fitzgerald, J., Haxthausen, A., Yenigun, H. (eds.) Theoretical Aspects of Computing - ICTAC 2008, Lecture Notes in Computer Science, vol. 5160, pp. 111–125. Springer Berlin / Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85762-4_8

[9] Deiters, C., Köster, M., Lange, S., Lützel, S., Mokbel, B., Mumme, C., (eds.), D.N.: DemSy - A Scenario for an Integrated Demonstrator in a SmartCity. Tech. Rep. 2010/01, NTH Focused Research School for IT Ecosystems, Clausthal University of Technology (May 2010), http://www.gbv.de/dms/clausthal/H_BIB/IfI/NTH_CompSciRep/2010-01.pdf

[10] Emerson, E., Halpern, J.: "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. Journal of the ACM (JACM) 33(1), 151–178 (1986)

[11] Enea, C., Dima, C.: Abstractions of multi-agent systems. In: Burkhard, H.D., Lindemann, G., Verbrugge, R., Varga, L. (eds.) Multi-Agent Systems and Applications V, Lecture Notes in Computer Science, vol. 4696, pp. 11–21. Springer Berlin / Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-75254-7_2

[12] Halpern, J., Fagin, R.: Modelling knowledge and action in distributed systems. Distributed computing 3(4), 159–177 (1989)

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

[14] Henzinger, T.A., Jhala, R., Majumdar, R.: Counterexample-guided control. In: ICALP. Lecture Notes in Computer Science, vol. 2719, pp. 886–902. Springer-Verlag (2003)

[15] Henzinger, T.A., Majumdar, R., Mang, F.Y.C., Raskin, J.F.: Abstract interpretation of game properties. In: Proceedings of the 7th International Symposium on Static Analysis. pp. 220–239. SAS '00, Springer-Verlag, London, UK (2000), http://portal.acm.org/citation.cfm?id=647169.718154

[16] Jamroga, W., Ågotnes, T.: Modular interpreted systems: A preliminary report. Tech. Rep. IfI-06-15, Clausthal University of Technology (2006)

[17] Jamroga, W., Ågotnes, T.: Modular interpreted systems. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) AAMAS. p. 131. IFAAMAS (2007)

[18] Kupferman, O., Vardi, M.Y.: An automata-theoretic approach to modular model checking. ACM Trans. Program. Lang. Syst. 22, 87–128 (January 2000), http://doi.acm.org/10.1145/345099.345104

[19] Kurshan, R.: Computer-aided verification of coordinating processes: the automata-theoretic approach. Princeton Univ Press (1994)

[20] McMillan, K.: Applying sat methods in unbounded symbolic model checking. In: Brinksma, E., Larsen, K. (eds.) Computer Aided Verification, Lecture Notes in Computer Science, vol. 2404, pp. 303–323. Springer Berlin / Heidelberg (2002), http://dx.doi.org/10.1007/3-540-45657-0_19

[21] McMillan, K.: Symbolic model checking: An Approach to the State Explosion Problem. Kluwer Academic Publishers Norwell, MA, USA (1993)

[22] Wooldridge, M.: Computationally grounded theories of agency. Multi-Agent Systems, International Conference on 0, 0013 (2000)