Abstraction in model checking multi-agent systems

Mika Cohen
Department of Computing
Imperial College London
London, UK

Mads Dam
Access Linnaeus Center
Royal Institute of Technology
Stockholm, Sweden

Alessio Lomuscio
Department of Computing
Imperial College London
London, UK

Francesco Russo
Department of Computing
Imperial College London
London, UK

ABSTRACT

We present an abstraction technique for multi-agent systems preserving temporal-epistemic specifications. We abstract a multi-agent system, defined in the interpreted systems framework, by collapsing the local states and actions of each agent in the system. We show that the resulting abstract system simulates the concrete system, from which we obtain a preservation theorem: If a temporal-epistemic specification holds on the abstract system, the specification also holds on the concrete one. In principle this permits us to model check the abstract system rather than the concrete one, thereby saving time and space in the verification step. We illustrate the abstraction technique with two examples. The first example, a card game, illustrates the potential savings in the cost of model checking a typical MAS scenario. In the second example, the abstraction technique is used to verify a communication protocol with an arbitrarily large data domain.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Model checking; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Modal logic

General Terms

Verification, theory

Keywords

Model checking, abstraction, epistemic logic

1. INTRODUCTION

Model checking [4] is a well-established automated technique for verifying reactive systems against design requirements expressed in temporal logics. More recently, model checking has been extended to multi-agent systems (MAS) and design requirements specified in agent logics, which extend temporal logics with agent-related modalities, such as epistemic modalities (cf. [2, 13, 16, 18, 20]).

Cite as: Abstraction in model checking multi-agent systems, , *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

A major obstacle to model checking is the so called *state explosion problem*. Since model checking involves a search through the state space of the system to be analyzed, model checking becomes intractable for systems with sufficiently large state spaces. In fact, relatively small system descriptions ("programs") can induce intractable state spaces.

Often, however, some of the details of a large, intractable system are irrelevant to the property we aim to verify. For reactive systems, there are *abstraction* techniques [6, 7] to simplify a system into a smaller, "abstract", system such that the temporal property to be verified holds for the original system if it holds for the abstract system. The latter can be checked automatically, if the abstract system is sufficiently small.

In this paper, we extend existential abstraction [6] from reactive systems to multi-agent systems, specified in the mainstream framework of *interpreted systems* [12]. Specifically, we reduce the state space generated by a given multi-agent system by simplifying each agent in the system. We show that if the reduced multi-agent system satisfies the property to be verified − given as a formula in temporal-epistemic logic − so does the original system.

We abstract a multi-agent system by simplifying the local states, the local protocol and the local evolution function of each agent. These simplifications alone suffice to guarantee that the epistemic possibility relations (the Kripkean accessibility relations for the epistemic modalities) in the abstract system simulate the epistemic possibility relations in the original system. Since the epistemic accessibility relations are not reduced directly, the abstraction technique proposed here has the same computational cost for temporal-epistemic specifications as for purely temporal specifications.

We illustrate the abstraction technique with two examples. The first example, a card game, illustrates the potential savings in the cost of model checking a typical MAS scenario. In the second example, the abstraction technique is used to verify a communication protocol with an arbitrarily large data domain.

Related work.

Abstraction of reactive systems for temporal properties is an active research area, with considerable attention dedicated to automatic techniques for building and refining abstract systems in particular programming environments (cf. [1, 5, 8, 15]). However, abstraction of multi-agent systems has so far received little attention.

Abstraction for epistemic properties has been attempted

before in [9, 11]. The abstraction technique in [9] has the novel feature of allowing several agent names in the original model to be reduced to the same abstract name in the abstract model. However, the models in [9, 11] are not computationally grounded [19], which, as argued in [19], may hamper the application of the technique to any concrete scenario. The models are simply arbitrary Kripke models with one primitive accessibility relation for each epistemic modality in the logic; to abstract a model one computes approximating relations for each epistemic accessibility relation. Even if the original Kripke model is induced by a multi-agent system, the abstraction technique does not produce another, smaller multi-agent system that can be fed into a model checker.

Outline of the paper.

The rest of the paper is organized as follows. Section 2 recalls the interpreted systems framework and Section 3 the temporal-epistemic specification logic ACTLK. Section 4 introduces existential abstraction of interpreted systems, and section 5 discusses simulation between interpreted systems. Section 6 shows that abstract interpreted systems simulate concrete interpreted systems; hence, ACTLK properties are preserved in the process. Section 7 illustrates abstraction of interpreted systems on a card game, while Section 8 looks at abstraction for a transmission protocol. Finally, Section 9 concludes and considers future work.

2. INTERPRETED SYSTEMS

We model multi-agent systems in the *interpreted systems framework* [12], where each agent is described by a set of possible local states it can be in, a set of actions it can perform, a local protocol selecting actions depending on the current local state, and a local evolution function specifying how an agent evolves from one local state to another depending on its own action and the actions of other agents.[1]

DEFINITION 2.1 (INTERPRETED SYSTEM). *An interpreted system over a set Ag of $n > 0$ agents and a set A of propositions ("atoms") is a tuple*

$$\mathcal{I} = \langle \{L_i\}_{i\in Ag}, \{ACT_i\}_{i\in Ag}, \{P_i\}_{i\in Ag}, \{t_i\}_{i\in Ag}, I_0, V\rangle$$

where:

- *L_i is a non-empty set of possible local states for agent i. The set of possible global states is $S = L_1 \times \cdots \times L_n$. For any global state $g \in S$, we write g_i for the i-th component in g, i.e., the local state of agent i in g.*
- *ACT_i is a non-empty set of possible actions for agent i. The set of possible joint actions is $ACT = ACT_1 \times \cdots \times ACT_n$.*
- *$P_i \subseteq L_i \times ACT_i$ is the local protocol for agent i.*
- *$t_i \subseteq L_i \times ACT \times L_i$ is the local evolution function for agent i.*[2]
- *$I_0 \subseteq S$ is a non-empty set of initial states.*
- *$V : S \longrightarrow 2^A$ is the evaluation function for propositions.*

The local protocols and the local evolution functions together determine how the system of agents proceeds from one global state to the next. Informally, the system can move from global state g to global state g' in one step if there is a joint action possible at g which transforms each local state g_i into g'_i.

DEFINITION 2.2 (GLOBAL TRANSITION RELATION). *The global transition relation in $\mathcal{I}$ is the relation $R \subseteq S \times S$ such that $\langle g, g'\rangle \in R$ if and only if:*

$$\exists \overline{a} \in ACT : \forall i \in Ag : \langle g_i, \overline{a}, g'_i\rangle \in t_i \ \wedge \ \langle g_i, a_i\rangle \in P_i$$

We assume that the global transition relation R always is serial, i.e., for every $g \in S$, there is $g' \in S$ such that gRg'.

DEFINITION 2.3 (PATH). *A path in $\mathcal{I}$ is an infinite sequence $g^0, g^1, \ldots$ of global states in S such that every pair of adjacent states forms a transition, i.e., gRg^{i+1} for all i.*[3]

A global state is reachable if there is a path leading to it from an initial state:

DEFINITION 2.4 (REACHABLE STATES). *The set G of reachable states contains all global states $g \in S$ for which there is a path $g^0, g^1, \ldots, g, \ldots$ starting from an initial state $g^0 \in I_0$.*

Intuitively, the local state of an agent contains all the evidence available to that agent: If $g_i = g'_i$ then system state g could, for all agent i can tell, be system state g'.

DEFINITION 2.5 (EPISTEMIC POSSIBILITY). *The epistemic possibility relation for agent i in system $\mathcal{I}$ is:*

$$\sim_i \ = \ \{\langle g, g'\rangle \in G \times G \mid g_i = g'_i\}$$

3. SPECIFICATION LOGIC ACTLK

We consider specifications ("system requirements") expressed in the logic ACTLK [17], which adds epistemic modalities to the temporal logic ACTL, the universal fragment of Computation Tree Logic [10].

DEFINITION 3.1 (ACTLK). *ACTLK formulae over a set Ag of agents and a set A of propositions are defined by:*

$$\phi ::= \alpha \mid \neg\alpha \mid \phi \wedge \phi \mid \phi \vee \phi \mid K_i\phi \mid AX\phi \mid A(\phi U \phi) \mid A(\phi R \phi)$$

where $\alpha \in A$ and $i \in Ag$.

The epistemic modality K_i is read "Agent i knows that" or "Agent i has the information to conclude that", the symbol A is read "For all paths", the symbol X is read "In the next state", the symbol U is read "Until" and, finally, the symbol R is read "Releases". Thus, the formula $AX\phi$ is read "For all paths, in the next state, ϕ"; the formula $A(\phi U \phi')$ is read "For all paths, ϕ holds until ϕ' holds"; the formula $A(\phi R \phi')$ is read "For all paths, ϕ releases ϕ'". We assume abbreviations customary for ACTL: $\top$ is any propositional logic tautology, $\bot$ is $\neg\top$, $AF\phi$ is $A(\top U \phi)$ ("For all paths, eventually ϕ") and $AG\phi$ is $A(\bot R \phi)$ ("For all paths, always ϕ").

[1] We follow [18] and assume a local evolution function for each agent . For ease of presentation, we omit the *environment*, which usually forms part of an interpreted system. It can be added with no difficulty.

[2] The local transition function t_i can be rewritten as a function $ACT \longrightarrow (L_i \times L_i)$.

[3] Note that we write g^i for the i:th global state in a given sequence of global states, while we use g_i for the local state of agent i inside the global state g.

As has been noted before, the combination of temporal and epistemic modalities allows us to specify, among other properties, what agents in a system are expected to know, and expected to know about what other agents know, and how such knowledge evolves over time. For example, $AG(i\,sent\,m\,to\,j \rightarrow AF(K_iK_j i\,sent\,m\,to\,j))$ expresses that whenever agent i sends a (message) m to agent j, then eventually agent i will know that agent j knows that i did so.

Given an interpreted system $\mathcal{I}$, the ACTL modalities are interpreted by means of the global transition relation R, while the epistemic modality K_i is interpreted by the epistemic possibility relation $\sim_i$ of agent i:

DEFINITION 3.2 (SATISFACTION). *Let $\mathcal{I}$ be an interpreted system over the set Ag of agents and the set A of propositions, let ϕ be an ACTLK formula over Ag and A, and let $g \in G$ be a reachable state. Truth of ϕ at g in $\mathcal{I}$, written $(\mathcal{I}, g) \models \phi$, is defined inductively by the following conditions:*

- $(\mathcal{I}, g) \models \alpha$ *iff* $\alpha \in V(g)$, *for* $\alpha \in A$
- $(\mathcal{I}, g) \models \neg\alpha$ *iff* $(\mathcal{I}, g) \not\models \alpha$
- $(\mathcal{I}, g) \models \phi \wedge \phi'$ *iff* $(\mathcal{I}, g) \models \phi$ *and* $(\mathcal{I}, g) \models \phi'$
- $(\mathcal{I}, g) \models \phi \vee \phi'$ *iff* $(\mathcal{I}, g) \models \phi$ *or* $(\mathcal{I}, g) \models \phi'$
- $(\mathcal{I}, g) \models K_i\phi$ *iff* $(\mathcal{I}, g') \models \phi$ *for all* g' *such that* $g \sim_i g'$
- $(\mathcal{I}, g) \models AX\phi$ *iff for every path* $g^0, g^1, \ldots$ *in* $\mathcal{I}$ *such that* $g = g^0$, *we have* $(\mathcal{I}, g^1) \models \phi$
- $(\mathcal{I}, g) \models A(\phi U \phi')$ *iff for every path* $g^0, g^1, \ldots$ *in* $\mathcal{I}$ *such that* $g = g^0$, *there is a natural number* i *such that* $(\mathcal{I}, g^i) \models \phi'$ *and* $(\mathcal{I}, g^j) \models \phi$ *for all* $0 \leq j < i$
- $(\mathcal{I}, g) \models A(\phi R \phi')$ *iff for every* i *and every path* $g^0, g^1, \ldots$ *in* $\mathcal{I}$ *such that* $g = g^0$, *if for all* $0 \leq j < i$, $(\mathcal{I}, g^j) \not\models \phi$ *then* $(\mathcal{I}, g^i) \models \phi'$

Formula ϕ is true in $\mathcal{I}$, $\mathcal{I} \models \phi$, iff $(\mathcal{I}, g) \models \phi$ for all $g \in I_0$.

4. EXISTENTIAL ABSTRACTION OF INTERPRETED SYSTEMS

In systems with large state spaces, it is infeasible to verify design requirements by considering the reachable states, even if represented symbolically [3]. Often however, not all aspects of system states are relevant to a given design requirement. If so, it seems natural to reduce the state space to a manageable size by simplifying states, removing irrelevant aspects. In existential abstraction [6], one reduces a large, possibly infinite reactive system − referred to as the *concrete* system − into a smaller reactive system − referred to as the *abstract* system − by partitioning the system states into equivalence classes. Each equivalence class, called an *abstract state*, forms a state in the abstract system. The abstract state is labelled with propositions that all concrete states in that class agree on. For every transition between states in the concrete system, one provides a corresponding transition between the respective equivalence classes in the abstract system: every behavior of the concrete system is a behavior also of the abstract system.

Here, we extend existential abstraction to interpreted systems by abstracting each agent i separately. We partition the set L_i of possible local states of agent i into equivalence classes; each equivalence class, called an *abstract local state of agent i*, forms a possible local state of agent i in the abstract system. Similarly, we partition the set ACT_i of possible actions of agent i into equivalence classes; each equivalence class, called an *abstract action* of agent i, is a possible action of agent i in the abstract system.

Local protocols and and local evolution functions are abstracted by uniformly replacing any local state l with its equivalence class $[l]$, and uniformly replacing any action a with its equivalence class $[a]$. Thus, if the local protocol of agent i in the concrete system selects an action a at a local state l, then the local protocol of i in the abstract system selects the abstract action $[a]$ at the abstract local state $[l]$. Analogously, if the local evolution function of agent i in the concrete system transforms a local state l into a local state l' under a joint action $\langle a_1, \ldots, a_n \rangle$, then the local evolution function of agent i in the abstract system transforms the abstract local state $[l]$ into the abstract local state $[l']$ under the joint abstract action $\langle [a_1], \ldots, [a_n] \rangle$.

Finally, we abstract the evaluation function by removing propositions that distinguish between equivalent local states; for propositions that remain, we let any abstract global state $\langle [g_1], \ldots, [g_n] \rangle$ inherit the propositions of the concrete global state g.

Formally, the abstract interpreted system is formed as a quotient construction as follows.

DEFINITION 4.1 (QUOTIENT OF INTERPRETED SYSTEM). *Assume an interpreted system $\mathcal{I}$ over the set Ag of agents and the set A of propositions. For each $i \in Ag$, assume an equivalence $\equiv_i \subseteq L_i \times L_i$ and an equivalence $\equiv_i \subseteq ACT_i \times ACT_i$. For $l \in L_i$, write $[l]$ for the equivalence class of l with respect to $\equiv_i$. Similarly, write $[a]$ for the equivalence class of $a \in ACT_i$ with respect to $\equiv_i$. Write $[g]$ for $\langle [g_1], \ldots, [g_n] \rangle$ and write $[\overline{a}]$ for $\langle [a_1], \ldots, [a_n] \rangle$. Let $A' \subseteq A$ consist of all propositions of A that do not distinguish between equivalent local states, i.e., all $\alpha \in A$ such that for all $g, g' \in S$: if $\alpha \in V(g)$ and $g_i \equiv g'_i$ for all $i \in Ag$, then $\alpha \in V(g')$.*

The quotient system of $\mathcal{I}$ is the interpreted system $\mathcal{I}'$ over the set Ag of agents and the set A' of proposition such that:

1. $L'_i = \{[l] \mid l \in L_i\}$.
2. $ACT'_i = \{[a] \mid a \in ACT_i\}$.
3. $P'_i = \{\langle [l], [a] \rangle \mid \langle l, a \rangle \in P_i\}$.
4. $t'_i = \{\langle [l], [\overline{a}], [l'] \rangle \mid \langle l, \overline{a}, l' \rangle \in t_i\}$.
5. $I'_0 = \{[g] \mid g \in I_0\}$.
6. $V'([g]) = V(g) \cap A'$.

Definition 4.1 does not say how the equivalence relations $\equiv_i$ are chosen. This issue is addressed in sections 6 to 8 below. The important property of quotient systems, however, is that specifications are preserved from abstract systems to concrete ones, as we show below.

5. SIMULATION

There is a standard notion of simulation for reactive systems [6]: A system simulates another system if every behavior of the latter is a behavior of the former. Since ACTL operators quantify over all behaviors (paths), it follows that any ACTL property that holds in the simulating system holds also in the simulated system. To extend this property preservation to ACTLK, we require that any epistemic

possibility in the simulated system can be "matched" by an epistemic possibility in the simulating system.

DEFINITION 5.1 (SIMULATION). *Assume an interpreted system $\mathcal{I}$ over the set Ag of agents and the set A of propositions, and an interpreted system $\mathcal{I}'$ over the same set Ag of agents and a subset $A' \subseteq A$ of propositions. A simulation relation between $\mathcal{I}$ and $\mathcal{I}'$ is a relation $\simeq \subseteq S \times S'$ such that:*

1. If $g \in I_0$ then $g' \in I'_0$ for some g' such that $g \simeq g'$

and whenever $g \simeq g'$ then:

2 $V'(g') = V(g) \cap A'$

3 If gRs then $g'R's'$ for some s' such that $s \simeq s'$

4 If $g \sim_i s$ then $g' \sim_i s'$ for some s' such that $s \simeq s'$

where R and R' are the global transition relations in, respectively, $\mathcal{I}$ and $\mathcal{I}'$. If there is a simulation relation between $\mathcal{I}$ and $\mathcal{I}'$, we say that $\mathcal{I}'$ simulates $\mathcal{I}$.

According to (1), every initial state in $\mathcal{I}$ can be matched by an initial state in $\mathcal{I}'$. According to (2), related states must agree on propositions in A'. Informally, A' contains propositions from A that we consider "relevant". According to (3), every transition in $\mathcal{I}$ can be matched by a transition in $\mathcal{I}'$. Any ACTLK property is preserved from the simulating system $\mathcal{I}'$ to the system $\mathcal{I}$ being simulated:

LEMMA 5.2. *Assume $\mathcal{I}'$ simulates $\mathcal{I}$. For any ACTLK formula ϕ over A', if $\models_{\mathcal{I}'} \phi$, then $\models_{\mathcal{I}} \phi$.*

PROOF (SKETCH). Assume a simulation relation $\simeq$ between $\mathcal{I}$ and $\mathcal{I}'$. We show that

$$g \simeq g',\ g' \models_{\mathcal{I}'} \phi \quad \Longrightarrow \quad g \models_{\mathcal{I}} \phi \tag{1}$$

by induction over ϕ. Base step, ϕ is α or $\neg\alpha$ for some $\alpha \in A$: From simulation requirement 2. Induction step, CTL modalities: From simulation requirement 3. For details cf. [6]. Induction step, epistemic modality: From simulation requirement 4. The Lemma follows from (1) and simulation requirement 1. □

6. PRESERVATION THEOREM

When we collapse local states and actions in an interpreted system, the resulting abstract system simulates the original system; every behavior and every epistemic possibility of the original system can be matched by a behavior and an epistemic possibility of the abstract system.

LEMMA 6.1. *If $\mathcal{I}'$ is a quotient of $\mathcal{I}$, then $\mathcal{I}'$ simulates $\mathcal{I}$.*

PROOF. We show that the relation $\simeq = \{\langle g, [g]\rangle \mid g \in S\}$ is a simulation between $\mathcal{I}$ and $\mathcal{I}'$. Simulation requirement 1: From requirement 5 in Definition 4.1. Simulation requirement 2: From requirement 6 in Definition 4.1. Simulation requirement 3: Assume gRs. By Definition 2.2, there is a $\overline{a} \in ACT$ such that (for all $i \in Ag$): $\langle g_i, \overline{a}, s_i\rangle \in t_i$ and $\langle g_i, a_i\rangle \in P_i$. By requirements 3 and 4 in Definition 4.1, $\langle [g_i], [\overline{a}], [s_i]\rangle \in t'_i$ and $\langle [g_i], [a_i]\rangle \in P'_i$, i.e., $\langle [g]_i, [\overline{a}], [s]_i\rangle \in t'_i$ and $\langle [g]_i, [\overline{a}]_i\rangle \in P'_i$, i.e., by Definition 2.2, $[g]R'[s]$. Thus,

$$gRs \quad \Longrightarrow \quad [g]R'[s] \tag{2}$$

from which simulation requirement 3 follows. Simulation requirement 4: Assume $g \sim_i s$. By Definition 2.5, $g_i = s_i$, i.e., $[g_i] = [s_i]$, i.e.,

$$[g]_i = [s]_i \tag{3}$$

Also by definition 2.5, $g, s \in G$. By (2) and simulation requirement 1, $[g], [s] \in G'$. Thus, simulation requirement 4 follows by (3). □

Since the abstract system simulates the concrete system, design requirements expressed in ACTLK are preserved from the abstract system to the concrete system.

THEOREM 6.2 (PRESERVATION). *Let $\mathcal{I}'$ be a quotient of interpreted system $\mathcal{I}$. For any ACTLK formula ϕ over A', if $\models_{\mathcal{I}'} \phi$, then $\models_{\mathcal{I}} \phi$.*

PROOF. From Lemma 5.2 and Lemma 6.1. □

Thus, if we have a multi-agent system $\mathcal{I}$ which is too large to model check, we can instead check the quotient system $\mathcal{I}'$. If the model checker reports that the specification ϕ holds, Theorem 6.2 allows us to conclude that ϕ holds also for the original system $\mathcal{I}$.

When applying the theorem, the main challenge is how to choose the collapsing equivalence relations $\equiv_i$ on local states and actions. One approach, following predicate abstraction [14], would be to select a finite set of local state predicates, and identify any two local states satisfying exactly the same predicates. Alternatively, following data abstraction [6], we may collapse local states by collapsing the data values they are built from. We illustrate this approach in Section 8.

7. A CARD GAME EXAMPLE

We illustrate the abstraction technique presented above on a simple card game. Two players, A and B, receive 9 cards each from a deck of 20 cards. The deck contains 10 red cards, $r_1, \ldots, r_{10}$, and 10 black cards, $b_1, \ldots, b_{10}$. Every red card beats every black card, otherwise higher indexed cards beat lower index cards. In each round of the game, each player plays a card from her hand. The better of the cards played wins the round. The game continues until all cards have been played. The player who won the most number of rounds wins the game.

We model the game as an interpreted system with three agents: the two players A and B and a score keeper S. Let $\mathcal{C}$ be the set of cards. We proceed to define, for each agent $i \in Ag = \{A, B, S\}$, its set of possible actions ACT_i, its set of possible local states L_i, its local protocol P_i and its local evolution function t_i.

A player can either play a card or do nothing (ϵ); the set ACT_i of actions for player $i \in \{A, B\}$ is:

$$ACT_i \quad = \quad \{play\, c \mid c \in \mathcal{C}\} \cup \{\epsilon\}$$

The score keeper can either evaluate who wins the round or do nothing:

$$ACT_S \quad = \quad \{eval, \epsilon\}$$

The local state of an agent contains the "information" available to her; in this game, a player $i \in \{A, B\}$ sees her hand and remembers the moves played so far:

$$L_i \quad = \quad \{\langle h, m\rangle \in 2^{\mathcal{C}} \times ACT^* \mid\ |h| + |m| = 9\}$$

where $h \subseteq \mathcal{C}$ represents the current hand of the agent and $m \in ACT^*$ represents the remembered sequence of joint actions. The score keeper, we assume, just keeps a record of the score so far:

$$L_S \quad = \quad \{\langle a, b\rangle \in \{0..9\} \times \{0..9\} \mid\ a + b \leq 9\}$$

where the record $\langle a, b\rangle$ says that player A has won a number of rounds and player B has won b number of rounds.

The local protocol P_i selects actions for agent i depending on the current local state of i; here, a player $i \in \{A, B\}$ plays a card randomly from its hand until it has no cards left:[4]

$$\begin{aligned} P_i(\langle h, m\rangle) &= \{play\, c \mid c \in h\}, \text{ if } h \neq \emptyset \\ P_i(\langle h, m\rangle) &= \{\epsilon\}, \text{ if } h = \emptyset \end{aligned}$$

The score keeper evaluates the score of each round until all cards have been played:

$$\begin{aligned} P_S(\langle a, b\rangle) &= \{eval\}, \text{ if } a + b < 9 \\ P_S(\langle a, b\rangle) &= \{\epsilon\}, \text{ if } a + b = 9 \end{aligned}$$

The local evolution function t_A for player A, specifying how the local state of player A is updated by a joint action, contains the following transitions (where $l \xrightarrow{\overline{a}} l'$ is the alternative notation for the triple $\langle l, \overline{a}, l'\rangle$, and $\cdot$ is the append operation on sequences):

$$\langle h, m\rangle \xrightarrow{\langle play\, c, play\, c', eval\rangle} \langle h \setminus \{c\}, m \cdot \langle play\, c, play\, c', eval\rangle\rangle$$

$$\langle h, m\rangle \xrightarrow{\langle \epsilon, \epsilon, \epsilon\rangle} \langle h, m\rangle$$

In the first transition above, the card c played by A is removed from the hand h and the move $\langle play\, c, play\, c', eval\rangle$ is appended to the memory m. In the second transition, the local state is unchanged when the agents do nothing. The local evolution function t_B for player B is defined in the same way as t_A above, but with the hand $h \setminus \{c'\}$ in the successor state in the first transition.

The score keeper adds a point to the player who plays the strongest card. Formally, the local evolution function t_S for the score keeper contains the following transitions:

$$\langle a, b\rangle \xrightarrow{\langle play\, c, play\, c', eval\rangle} \langle a + 1, b\rangle, \text{ if } c > c'$$

$$\langle a, b\rangle \xrightarrow{\langle play\, c, play\, c', eval\rangle} \langle a, b + 1\rangle, \text{ if } c' > c$$

$$\langle a, b\rangle \xrightarrow{\langle \epsilon, \epsilon, \epsilon\rangle} \langle a, b\rangle$$

where $>$ is the total ordering of cards (i.e., $r_i > b_j$, and if $i > j$ then $r_i > r_j$ and $b_i > b_j$). In the first transition above, A plays the stronger card and so the score keeper adds a point to the score of A, and analogously for the second transition.

To complete the interpreted system for the card game, we define the set I_0 of initial states and the evaluation function V. Initially, each player holds 9 unique cards, has no moves recorded and has 0 points: I_0 is the set of all global states $\langle\langle h, m\rangle, \langle h', m'\rangle, \langle a, b\rangle\rangle$ such that

$$|h| = |h'| = 9,\ h \cap h' = \emptyset,\ |m| = |m'| = 0,\ \ a = b = 0$$

We assume the set A contains the propositions $onlyred_i$ ("Player i holds only red cards.") and win_i ("Player i has won the game."), for $i \in \{A, B\}$. The evaluation function V interprets the propositions as expected:

$$\begin{aligned} onlyred_A \in V(\langle\langle h, m\rangle, l, l'\rangle) &\Leftrightarrow h \cap \{b_1, \ldots, b_{10}\} = \emptyset \\ win_A \in V(\langle l, l', \langle a, b\rangle\rangle) &\Leftrightarrow a > b \text{ and } a + b = 9 \end{aligned}$$

[4] Here we follow standard practice for interpreted systems and treat P_i as a function $L_i \longrightarrow 2^{ACT_i}$, rather than as a relation as in earlier sections. Formally, we write $P_i(l) = X$ when $X = \{a \mid \langle l, a\rangle \in P_i\}$.

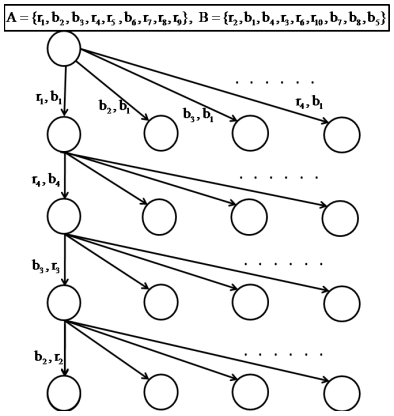

Figure 1: Sketch of the global transition relation for the concrete card game.

The conditions for the propositions $onlyred_B$ and win_B are analogous. This completes the definition of the interpreted system $\mathcal{I}$ for the card game.

Figure 1 sketches the resulting global transition relation R, following it from one particular initial state - In total, there are approximately 9 million possible initial states. For the sake of readability, we label transitions in the figure with joint actions; the label c, c' abbreviates the joint action $\langle play\, c, play\, c', eval\rangle$. All nodes have transitions to other nodes, even if not shown in the figure. The dots indicate omitted transitions.

It is quite clear that if one player is served only red cards she will win the game whatever the moves played. In fact we can verify this and even stronger temporal epistemic specifications. For example, we could try to model check whether

$$onlyred_B \rightarrow K_B(AF\, win_B \wedge K_A AF\, win_B) \qquad (4)$$

The formula states that if B is served only red cards then not only does she know that she will win the game but also she knows that A knows this.

It turns out, however, that the state space is too large for the system to be checked even by symbolic techniques such as BDDs [3]. But, instead of checking the specification above directly, we can first abstract the system to reduce its state space as we show below.

We collapse all red cards into one, collapse all black cards into one and ignore the memory ("action log") m of players. Let $\rho : \mathcal{C} \longrightarrow \{red, black\}$ map all red cards to the abstract card red and all black cards to the abstract card $black$, i.e., $\rho(r_i) = red$ and $\rho(b_i) = black$. For each local state $\langle h, m\rangle$ of a player $i \in \{A, B\}$, we abstract away the memory m and the card indexes in the hand h:

$$\langle h, m\rangle \equiv_i \langle h', m'\rangle \quad \Leftrightarrow \quad \rho(h) = \rho(h')$$

where $\rho(h)$ is the multi-set $\{\rho(c) \mid c \in h\}$. For example, if $h = \{b_1, b_2, r_1\}$ then $\rho(h)$ is the multi-set $\{black, black, red\}$. We do not abstract the local states of the score keeper, so $\equiv_S$ is simply identity on L_S.

We abstract the actions of player $i \in \{A, B\}$ by removing the indexes from cards:[5]

$$play\, c \equiv_i play\, c' \quad \Leftrightarrow \quad \rho(c) = \rho(c') \qquad (5)$$

[5] To be precise, we take $\equiv_i$ to be the least equivalence on ACT_i satisfying the condition (5).

Again, we do not abstract the actions of the score keeper: $\equiv_S$ is identity on ACT_S.

We proceed to construct the abstract system $\mathcal{I}'$, i.e., the quotient of $\mathcal{I}$ with respect to equivalences $\equiv_A$, $\equiv_B$ and $\equiv_S$. Representing the abstract action $[play\, c]$ by $play\, \rho(c)$, and representing the abstract local state $[\langle h, m\rangle]$ by the multi-set $\rho(h)$, we have:

$$\begin{aligned} ACT_i' &= \{play\, red,\ play\, black,\ \epsilon\} \\ L_i' &= \{h \mid \text{ multi-set } h,\ |h| \leq 9\} \\ P_i(h) &= \{play\, x \mid x \in h\},\ \text{if } h \neq \emptyset \\ P_i(h) &= \{\epsilon\},\ \text{if } h = \emptyset \end{aligned}$$

for players $i \in \{A, B\}$, while $ACT_S' = ACT_S$, $L_S' = L_S$ and $P_S' = P_S$.

The abstract local evolution function t_A' for player A contains the following transitions:

$$\begin{array}{lcl} h & \xrightarrow{\langle play\, x, play\, x', eval\rangle} & h \setminus \{x\} \\ h & \xrightarrow{\langle \epsilon, \epsilon, \epsilon\rangle} & h \end{array}$$

for abstract cards $x, x' \in \{red, black\}$, where $h \setminus \{x\}$ is the result of removing one instance of the abstract card x from the multi-set h. The abstract local evolution function t_B' for player B is defined in the same way, but with the multi-set $h \setminus \{x'\}$ as the successor state in the first transition.

The abstract local evolution function t_S' for the score keeper contains the following transitions:

$$\begin{array}{lcl} \langle a, b\rangle & \xrightarrow{\langle play\, x, play\, x,\, eval\rangle} & \langle a+1, b\rangle \\ \langle a, b\rangle & \xrightarrow{\langle play\, x, play\, x,\, eval\rangle} & \langle a, b+1\rangle \\ \langle a, b\rangle & \xrightarrow{\langle play\, red, play\, black,\, eval\rangle} & \langle a+1, b\rangle \\ \langle a, b\rangle & \xrightarrow{\langle play\, black, play\, red,\, eval\rangle} & \langle a, b+1\rangle \\ \langle a, b\rangle & \xrightarrow{\langle \epsilon, \epsilon, \epsilon\rangle} & \langle a, b\rangle \end{array}$$

for abstract card $x \in \{red, black\}$. In the first transition above, the score keeper adds a point to the score of A when players A and B play cards of the same color, while in the second transition, the score keeper adds a point to the score of B when players A and B play cards of the same color.

The set I_0' of abstract initial states consists of all abstract global states $\langle h,\, h',\, \langle a, b\rangle\rangle$ such that

$$\begin{aligned} |h| = |h'| = 9,\ a = b = 0 \\ |Red(h)| + |Red(h')| \leq 10,\ |Black(h)| + |Black(h')| \leq 10 \end{aligned}$$

where $Red(h)$ is the multi-set of red cards in multi-set h, and similarly for $Black(h)$.

The abstract and concrete systems have the same propositions, $A' = A$, since no proposition in A distinguishes between global states that have been identified. The abstract evaluation function V' is defined by:

$$\begin{aligned} onlyred_A \in V'(\langle h,\, l,\, l'\rangle) &\Leftrightarrow Black(h) = \emptyset \\ win_A \in V'(\langle l,\, l'\, \langle a, b\rangle\rangle) &\Leftrightarrow a > b \text{ and } a + b = 9 \end{aligned}$$

and analogously for propositions $onlyred_B$ and win_B.

This completes the construction of the abstract system $\mathcal{I}'$. Figure 2 depicts the resulting abstract global transition relation R', tracking it from one of the 36 possible initial states.

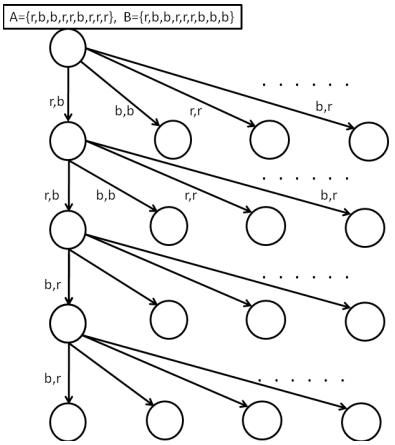

Figure 2: Scetch of the global transition relation for the abstract card game.

Having thus constructed the abstract system, we could feed it to a model checker and check wether the specification (4) - which uses only propositions from A' - holds. The model checker would report that the specification does indeed hold. By Preservation Theorem 6.2, we could then conclude that the specification holds also in the concrete system. Note, however, that checking a different specification might require a different abstraction. For instance, the formula $onlyred_B \rightarrow AF\, K_B\, win_B$ holds for the concrete system but not for the particular abstract system above.[6]

In this example, the abstraction technique reduces the number of reachable states that the model checker needs to compute by a factor of $5 \cdot 10^{11}$: The concrete system has approximately $3 \cdot 10^{18}$ reachable states, while the abstract system has approximately 5 million reachable states. Although the example is artificial and serves only as a vehicle to illustrate the presented abstraction technique, we expect state space savings of a similar magnitude in a number of scenarios that are amenable to abstraction.

8. TRANSMISSION PROTOCOL EXAMPLE

The bit transmission protocol [12] is a popular application of temporal-epistemic logic. In this protocol, a sender S wants to communicate some data to a receiver R over a faulty communication channel. When analyzing the protocol, one typically assumes – "for the sake of the example" – that the data communicated is just the value of a bit. Using abstraction, we can formally show that the data domain $\{0, 1\}$ indeed suffices: if the protocol achieves its goal for the data domain $\{0, 1\}$, it achieves its goal for any choice of data domain, no matter how large.

The transmission protocol for a data domain D runs as follows. A sender S and a receiver R communicate over a lossy channel. The goal of the protocol is to transmit a data value $d \in D$ from the sender S to the receiver R in such a way that the sender S will know that the receiver R knows the value d. The protocol specifies that S sends the data value to R, and continues to send it until S receives an acknowledgement from R. For its part, once R receives the data value, R sends an acknowledgement of receipt to S,

[6]We thank an anonymous reviewer for suggesting this formula to us.

and re-sends it indefinitely. When $|D| = 2$, the transmission protocol is the bit transmission protocol.

We implement the transmission protocol for a non-empty data domain D as an interpreted system $\mathcal{I}^D$ with two agents, the sender S and the receiver R. We assume the sender S observes her value and whether or not she has received an acknowledgement:

$$L_S \quad = \quad \bigcup_{d \in D} \{d, \langle d, ack \rangle\}$$

In local state d, the sender sees (has) her data value d, while in local state $\langle d, ack \rangle$, the sender sees (has) her value d and the acknowledgement. The receiver, on the other hand, either sees nothing or sees the data value:

$$L_R \quad = \quad \{\lambda\} \cup D$$

In local state λ, the receiver has not yet seen any value, while in local state $d \in D$, the receiver sees the value d.

The sender can send her data value or do nothing (ϵ):

$$ACT_S \quad = \quad \{send\, d \mid d \in D\} \cup \{\epsilon\}$$

while the receiver can send an acknowledgement of receipt of do nothing:

$$ACT_R \quad = \quad \{sendack\, d \mid d \in D\} \cup \{\epsilon\}$$

The protocol for the sender is to keep sending her data value until she receives an acknowledgement:

$$\begin{aligned} P_S(d) &= \{send\, d\} \\ P_S(\langle d, ack \rangle) &= \{\epsilon\} \end{aligned}$$

The receiver should do nothing until it received a data value, and then keep sending an acknowledgment:

$$\begin{aligned} P_R(\lambda) &= \{\epsilon\} \\ P_R(d) &= \{sendack\, d\} \end{aligned}$$

The local evolution function t_S for the sender contains the following transitions:

$$d \xrightarrow{\langle send\, d, \epsilon \rangle} d \tag{6}$$

$$d \xrightarrow{\langle send\, d, sendack\, d \rangle} d \tag{7}$$

$$d \xrightarrow{\langle send\, d, sendack\, d \rangle} \langle d, ack \rangle \tag{8}$$

$$\langle d, ack \rangle \xrightarrow{\langle \epsilon, sendack\, d \rangle} \langle d, ack \rangle \tag{9}$$

In (6), the joint action $\langle send\, d, \epsilon \rangle$ leaves the local state of S unchanged: Since the receiver does nothing, the sender obtains no new information. In (7), the receiver sends an acknowledgement which is lost on the communication channel, so the local state of S is unchanged. In (8), on the other hand, the acknowledgement reaches the sender, and the local state is updated accordingly. In (9), the sender stays in the same local state once the the sender has received the acknowledgement.

Analogously, the local evolution function t_R for the receiver contains the following transitions:

$$\begin{aligned} \lambda &\xrightarrow{\langle send\, d, \epsilon \rangle} \lambda \\ \lambda &\xrightarrow{\langle send\, d, \epsilon \rangle} d \\ d &\xrightarrow{\langle \pi, sendack\, d \rangle} d \end{aligned}$$

for $\pi \in ACT_S$.

Initially, the sender sees its data value d and the receiver sees nothing:

$$I_0 \quad = \quad \{\langle d, \lambda \rangle \mid d \in D\}$$

Let the set A of propositions contain the proposition *recack* ("The sender has received an acknowledgement") and the propositions $val = d$ ("The value is d"), for all $d \in D$. The evaluation function is as expected:

$$\begin{aligned} V(\langle d, l \rangle) &= \{val = d\} \\ V(\langle \langle d, ack \rangle, l \rangle) &= \{val = d,\ recack\} \end{aligned}$$

for $l \in L_R$. This completes the definition of the interpreted system $\mathcal{I}^D$ for the transmission protocol.

As discussed in the literature, suppose we would like to check wether whenever the sender S has obtained an acknowledgement, the sender S knows that the receiver R knows whether the value is d. Formally, we would like to determine, for all $d \in D$, if $\mathcal{I}^D$ satisfies:

$$AG(val = d \wedge reckack \rightarrow K_S K_R val = d) \tag{10}$$

Applying Preservation Theorem 6.2, we show that the specification (10) holds in $\mathcal{I}^D$ for any chosen data domain D, if the specification holds for the bit transmission protocol, i.e., if it holds for $D = \{0, 1\}$. The latter is, of course, feasible for a model checker to determine.

Fix any $d_0 \in D$. We abstract the concrete system $\mathcal{I}^D$ by identifying all data values d which are distinct from d_0. Define a data collapsing function $\rho : D \longrightarrow \{d_0, \neg d_0\}$ by:

$$\begin{aligned} \rho(d_0) &= d_0 \\ \rho(d) &= \neg d_0,\ if\ d \neq d_0 \end{aligned}$$

We identify local states if they are identical after applying ρ on the data values inside:

$$\begin{aligned} d \equiv_S d' &\quad \Leftrightarrow \quad \rho(d) = \rho(d') \\ d \equiv_R d' &\quad \Leftrightarrow \quad \rho(d) = \rho(d') \\ \langle d, ack \rangle \equiv_S \langle d', ack \rangle &\quad \Leftrightarrow \quad \rho(d) = \rho(d') \end{aligned}$$

Similarly, we identify actions that are identical after renaming:

$$\begin{aligned} send\, d \equiv_S send\, d' &\quad \Leftrightarrow \quad \rho(d) = \rho(d') \\ send\, ack\, d \equiv_R send\, ack\, d' &\quad \Leftrightarrow \quad \rho(d) = \rho(d') \end{aligned}$$

Now, let $\mathcal{I}'$ be the quotient of $\mathcal{I}^D$ with respect to equivalences $\equiv_S$ and $\equiv_R$.

If we represent the abstract local state $[l]$ by $\rho(l)$ (i.e, the result of substituting $\rho(d)$ for d in l), and represent the abstract action $[a]$ by $\rho(a)$, we see that the abstract system $\mathcal{I}'$ is just the bit transmission protocol model $\mathcal{I}^{\{d_0, \neg d_0\}}$, except that A' contains only the abstract propositions *reckack* and $val = d_0$.

If we feed the bit transmission protocol model $\mathcal{I}^{\{0,1\}}$ to a model checker, we will find that the specification (10) holds. From this we infer that the specification (10) holds also for the concrete system $\mathcal{I}^D$: Pick any $d_0 \in D$ and form the abstract system $\mathcal{I}'$ as above. As we have seen, the abstract system $\mathcal{I}'$ is just $\mathcal{I}^{\{0,1\}}$. So, by assumption, the abstract system $\mathcal{I}'$ satisfies:

$$AG(val = d_0 \wedge reckack \rightarrow K_S K_R val = d_0) \tag{11}$$

By Preservation Theorem 6.2, (11) holds for the concrete system $\mathcal{I}^D$. Since, d_0 was chosen arbitrarily from D, we can conclude the protocol goal (10) for all $d \in D$.

The transmission protocol (with variations) is a standard example used to illustrate data abstraction for reactive systems and temporal logic. The logical specifications usually considered specify only control flow, and therefore allow data to be abstracted away completely. The specification (10) here, by contrast, is about knowledge of data, and so does not allow data to be completely abstracted away; if we choose $\rho : D \longrightarrow \{0\}$, then A' in the abstract system will contain only the proposition *reckack*, and so Theorem 6.2 would not apply to the specification (10) .

9. CONCLUSION AND FUTURE WORK

Model checking MAS has received considerable attention in the MAS community. However, there has been little work so far on abstraction techniques for reducing the state space of MAS to a tractable size.

In this paper, we have presented an abstraction technique for MAS preserving temporal-epistemic properties. We have seen that when the state space of a MAS is too large for a model checker to compute, it is sometimes possible to reduce it by simplifying the local states and actions of agents before feeding the system to a model checker. If the model checker reports that a design requirement expressed in the temporal-epistemic logic ACTLK holds, we can conclude that the requirement holds also for the original system.

A natural next step for future research is to automate the abstraction process. We intend to attempt this for multi-agent programs in the *interpreted systems programming language* (ISPL) [18] by taking inspiration from predicate abstraction techniques used in reactive systems.

Acknowledgments.

The research described in this paper is partly supported by EPSRC funded project EP/E035655, by the European Commission Framework 6 funded project CONTRACT (IST Project Number 034418), and by grant 2003-6108 from the Swedish Research Council.

10. REFERENCES

[1] T. Ball and S. K. Rajamani. Boolean programs: A model and process for software analysis. MSR Technical Report 2000-14. 2000.

[2] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking AgentSpeak. In J. S. Rosenschein, T. Sandholm, W. Michael, and M. Yokoo, editors, *AAMAS-03*, pages 409–416. ACM Press, 2003.

[3] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.

[4] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs Workshop*, pages 52–71, London, UK, 1982. Springer-Verlag.

[5] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In E. A. Emerson and A. P. Sistla, editors, *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2000.

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

[7] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[8] S. Das and D. L. Dill. Successive approximation of abstract transition relations. In *LICS*, pages 51–60, 2001.

[9] F. Dechesne, S. Orzan, and Y. Wang. Refinement of kripke models for dynamics. In J. S. Fitzgerald, A. E. Haxthausen, and H. Yenigün, editors, *ICTAC*, volume 5160 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2008.

[10] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

[11] C. Enea and C. Dima. Abstractions of multi-agent systems. In H. Burkhard, G. Lindemann, R. Verbrugge, and L. Z. Varga, editors, *CEEMAS*, volume 4696 of *Lecture Notes in Computer Science*, pages 11–21. Springer, 2007.

[12] R. Fagin, J. Y. Halpern, M. Y. Vardi, and Y. Moses. *Reasoning about knowledge*. MIT Press, Cambridge, MA, USA, 1995.

[13] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *CAV'04*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

[14] S. Graf and H. Saïdi. Construction of abstract state graphs with pvs. In O. Grumberg, editor, *CAV*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1997.

[15] R. P. Kurshan. *Computer-aided verification of coordinating processes: the automata-theoretic approach*. Princeton University Press, Princeton, NJ, USA, 1994.

[16] W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, and M. Szreter. VerICS 2004: A model checker for real time and multi-agent systems. In *CS&P'04*, volume 170 of *Informatik-Berichte*, pages 88–99. Humboldt University, 2004.

[17] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[18] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. *Journal of Applied Logic*, 5(2):235–251, 2007.

[19] M. Wooldridge. Computationally grounded theories of agency. In E. Durfee, editor, *ICMAS*, pages 13–22. IEEE Press, 2000.

[20] M. Wooldridge, M. Fisher, M. Huget, and S. Parsons. Model checking multiagent systems with MABLE. In *AAMAS-02*, pages 952–959, Bologna, Italy, July 2002.