Science of Computer Programming 228 (2023) 102946

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Max-SAT-based synthesis of optimal and Nash equilibrium strategies for multi-agent systems

Nils Timm *, Josua Botha, Steven Jordaan

Department of Computer Science, University of Pretoria, Pretoria, South Africa

ARTICLE INFO

Article history:
Received 11 May 2022
Received in revised form 21 March 2023
Accepted 22 March 2023
Available online 28 March 2023

Keywords:
Bounded model checking
Multi-agent systems
Strategy synthesis
Max-SAT-based optimisation
Nash equilibrium

ABSTRACT

We present techniques for verifying strategic abilities of multi-agent systems via SAT-based and Max-SAT-based bounded model checking. In our approach we focus on systems of agents that pursue goals with regard to the allocation of shared resources. One of the problems to be solved is to determine whether a coalition of agents has a joint strategy that guarantees the achievement of all resource goals, irrespective of how the opposing agents in the system act. Our approach does not only decide whether such a winning strategy exists, but also synthesises the strategy.

Winning strategies are particularly useful in the presence of an opposition because they guarantee that each agent of the coalition will achieve its individual goal, no matter how the opposition behaves. However, for the grand coalition consisting of all agents in the system, following a winning strategy may involve an inefficient use of resources. A winning strategy will only ensure that each agent will reach its goal *at some time*. But in practical resource allocation problems it may be of additional importance that once-off resource goals will be achieved *as early as possible* or that repetitive goals will be achieved *as frequent as possible*. We present an extended technique that synthesises strategies that are *collectively optimal* with regard to such quantitative performance criteria.

A collectively optimal strategy allows to optimise the *overall* system performance but it may favour certain agents over others. In competitive scenarios a *Nash equilibrium* strategy may be a more adequate solution. It guarantees that no agent can improve its individual performance by unilaterally deviating from the strategy. We developed an algorithm that initially generates a collectively optimal strategy and then iteratively alternates this strategy until the strategy becomes a Nash equilibrium or a cycle of non-equilibrium strategies is detected.

Our approach is based on a propositional logic encoding of strategy synthesis problems. We reduce the synthesis of winning strategies to the Boolean satisfiability problem and the synthesis of optimal and Nash equilibrium strategies to the maximum satisfiability problem. Hence, efficient SAT- and Max-SAT solvers can be employed to solve the encoded strategy synthesis problems.

* Corresponding author.
E-mail address: ntimm@cs.up.ac.za (N. Timm).

https://doi.org/10.1016/j.scico.2023.102946

1. Introduction

Multi-agent systems for resource allocation (MRAs) have been introduced in [1] as a concept for modelling competitive resource allocation problems in distributed computing. An MRA is composed of a set of agents and a set of resources. Agents have access to a subset of the overall set of resources. Moreover, each agent has a goal in terms of the amount of resources to accumulate. Particular resources can be allocated by means of *request* actions. Further types of actions are *release* and *idle*. MRAs run in discrete rounds. In each round each agent selects an action, and the tuple of selected actions gets executed in a simultaneous manner. Once an agent has achieved its goal, it releases all accumulated resources and starts to allocate them again. Agents may pursue to achieve their goals once-off or repetitively. Since resources are generally shared, the achievement of goals is a competition between agents. Several practically relevant scenarios of resource allocation can be modelled as a multi-agent system for resource allocation. A real-world example is the resource competition in heterogeneous wireless sensor network (WSN) systems [2] used for industrial or environmental monitoring. In such systems different applications run on heterogeneous sensor nodes that belong to multiple WSNs. Each application has individual requirements such as receiving sensor data at a certain rate and via a certain delivery mode. Sensor nodes and communication channels belong to a shared infrastructure. Thus, the applications compete for these network resources. MRA-based modelling is an adequate paradigm for the simulation of heterogeneous WSN systems. Applications can be represented as agents, their requirements as goals, and sensor nodes as resources.

For MRAs (or more specifically, for the scenarios that they model) it is typically of importance that they are designed in a way such that for a certain group of agents the achievement of goals can be guaranteed, no matter how the remaining agents in the system may counter-act. We call such a group of agents a *coalition* and the remaining agents the *opposition*. Goal-achievability properties of a coalition A against an opposition B can be formalised in alternating-time logics such as ATL or ATL* [3], which extend classical temporal logics by strategic operators. An alternating-time formula $\langle\langle A\rangle\rangle\,\varphi$ expresses that the coalition A has a *strategy* to achieve φ, irrespective of how the opposition acts, where φ is a classical temporal logic formula. The formula φ characterises the collective goal of the coalition A, and it is composed of sub formulas that characterise the individual goals of the agents in A. In this context, a strategy is a mapping between states of the underlying MRA and actions to be taken by the agents in A in these states. The corresponding *coalition versus opposition model checking* problem is to determine whether such a winning strategy exists or not, and the *coalition versus opposition strategy synthesis* problem is to actually build the strategy. Established tools for deciding the PTIME-complete ATL model checking problem of general multi-agent systems such as Mocha [4] and MCMAS [5] are based on binary decision diagrams (BDDs). These tools primarily focus on the *verification* of multi-agent systems, but also provide support for *strategy synthesis*. Algorithms for the 2EXPTIME-complete ATL* model checking have been theoretically defined, but due to the high complexity no practically relevant implementation exists.

In [6] we introduced a SAT-based bounded model checking technique for verifying goal-achievability properties of multi-agent systems for resource allocation. Our technique does not only decide the coalition versus opposition model checking problem, it also synthesises a corresponding winning strategy if existent. The properties that we consider in our approach are specified in the alternating-time logic 1-ATL*. This logic is a subset of ATL* that is restricted to formulas with a single path quantifier and a single temporal operator. Our approach encodes the bounded model checking problem in propositional logic. Thus, model checking can be performed via satisfiability solving. From a satisfying truth assignment of the encoded problem a winning strategy for the coalition A can be immediately derived. For our bounded model checking problems linear completeness thresholds exist, which also makes unbounded model checking feasible. A distinct feature of our technique is our iterative strategy synthesis algorithm that allows to avoid the exhaustive consideration of all possible strategies for many practical instances.

Winning strategies are particularly useful in the presence of an opposition because they guarantee that each agent of the coalition will achieve its individual goal, no matter how the opposition acts. However, for the grand coalition consisting of all agents in the system, following a winning strategy may involve an inefficient use of resources. A winning strategy will only ensure that each agent will reach its goal *at some time*. But in practical resource allocation problems it may be of additional importance that once-off resource goals will be achieved *as early as possible* or that repetitive goals will be achieved *as frequently as possible*. For instance, the goal of a WSN-based application for fire detection is to receive certain environmental data – and it is essential that the data is received *at a high sample rate*. In this article, we extend our existing technique such that winning strategies can be synthesised that are *optimal* with regard to such quantitative criteria. For this, we augment our MRAs with goal-related pay-offs. A strategy is collectively optimal if it maximises the overall pay-off. The corresponding *optimal strategy synthesis problem* is no longer a pure decision problem and therefore cannot be reduced to standard Boolean satisfiability. We show that optimal strategy synthesis can be reduced to the *maximum satisfiability problem* (Max-SAT) [7]. Our reduction is based on an extension of our existing propositional logic encoding by *pay-off clauses*. We have proven that from a truth assignment that maximises the number of satisfied pay-off clauses a corresponding collectively optimal strategy can be immediately derived. This allows us to employ a Max-SAT solver [8] for synthesising winning strategies that ensure that agents will achieve their goals as early as possible or as frequently as possible.

A collectively optimal strategy ensures the qualitative objective that each agent will achieve its goal at least once, and it optimises the quantitative objective of interest such as the frequency of agents reaching their goal. The latter objective will only be optimised with regard to the *overall* multi-agent system but not necessarily with regard to each individual agent. For practical scenarios where just the overall system performance is of importance such a solution will be sufficient.

But for strictly competitive scenarios a collectively optimal strategy that favours certain agents (or WSN applications) over others may be not acceptable. For such scenarios a *Nash equilibrium strategy* [9] can be an adequate solution. We define a Nash equilibrium of an MRA as a strategy that ensures that: 1. Each agent will achieve its individual goal at least once. 2. No agent can increase its individual pay-off by deviating from the strategy with alternative strategic decisions, assuming that the remaining agents still adhere to the strategy. Hence, a Nash equilibrium is a strategy on which even competing agents may agree on. We developed an algorithm that initially generates a collectively optimal strategy and then iteratively alternates this strategy until the strategy becomes a Nash equilibrium or a cycle of non-equilibrium strategies is detected.

We have implemented the *coalition versus opposition* strategy synthesis on top of the solver CaDiCaL [10]. Moreover, the *collectively optimal* strategy synthesis and the *Nash equilibrium* synthesis have been implemented on top of the Max-SAT solver OPEN-WBO [8]. Experiments show promising performance results. To the best of our knowledge, our technique is the first Max-SAT-based approach to the synthesis of optimal an Nash equilibrium strategies for multi-agent systems.

The remainder of this article is organised as follows. Section 2 discusses related work. In Section 3 we introduce multi-agent systems for resource allocation, strategies, and the goal-achievability properties that we consider in our approach. Moreover, we define the coalition versus opposition strategy synthesis problem. Section 4 presents the propositional logic encoding of the coalition versus opposition strategy synthesis problem. In Section 5 we present the SAT-based algorithm for synthesising winning strategies that we originally introduced in [6]. In Section 6 we define the optimisation criteria frequency and speed. Moreover, we generalise the synthesis of winning strategies to the synthesis of collectively optimal and Nash equilibrium strategies. Section 7 presents the reduction of optimal and Nash equilibrium strategy synthesis to the maximum satisfiability problem. In Section 8 we introduce the implementation of our approach and we present experimental results. We conclude this article in Section 9 and give an outlook on future work.

2. Related work

Model checking has been originally introduced as a technique for verifying temporal logic properties of hardware and software designs [11]. Classical symbolic model checking approaches include BDD-based CTL model checking [12] and SAT-based bounded LTL model checking [13]. CTL model checking has been also extended to multi-agent systems [14]. While CTL and LTL do not consider strategic aspects, [3] introduced the alternating-time logics ATL and ATL*, which are logics for reasoning about strategies in multi-agent systems. The general ATL model checking problem is PTIME-complete whereas the ATL* model checking problem is 2EXPTIME-complete. Thus, while for ATL model checking efficient BDD tools like MCMAS [5] and MOCHA [4] exist, ATL* has been rather considered on a theoretical level [15]. SAT-based bounded model checking of multi-agent systems has been proposed in [16,17]. Similar to our technique, [16,17] unfold the transition relation k times by means of a propositional formula. However, their approaches are limited to the verification of epistemic properties and do not support strategic operators. These approaches have been only theoretically defined but not implemented. [18] presents a SAT-based unbounded ATL model checking technique. Although based on a reduction to SAT, this technique is very different from ours. In [18] a BDD-encoded model checking problem gets translated into a corresponding set of quantified Boolean formulas and fix-point equations, which can be further translated into a plain propositional encoding. [18] does not support strategy synthesis. An existing tool for synthesising ATL strategies is SMC [19]. SMC operates on a BDD model of the multi-agent system to be verified. It iteratively guesses a strategy, fixes the strategy in the model and checks whether it is a winning strategy, which reduces ATL model checking to CTL model checking in each iteration. Experimental results presented in [19] show that SMC can successfully synthesise strategies for systems with up to eight agents, which approximately matches with the capabilities of our SATMAS tool. While SMC is limited to the synthesis of winning strategies, our tool additionally provides support for the synthesis of optimal and Nash equilibrium strategies.

The synthesis of Nash equilibrium strategies for multi-agent systems has been studied in several works on *rational verification* [20–22]. Given a system where each agent has a qualitative goal and given a temporal logic formula φ, the rational verification problem is to determine whether a Nash equilibrium strategy exists that guarantees φ. In this context, the agents' goals and the temporal logic formula typically characterise different properties. A winning strategy must ensure φ but does not necessarily need to satisfy the goal of each agent. However, agents prefer strategies that satisfy their goal over those that do not. A joint winning strategy is a Nash equilibrium if no agent that under this strategy fails to achieve its goal can come up with an alternative individual strategy that will enable the agent to succeed in achieving its goal. Hence, in rational verification the goals are entirely qualitative. In contrast, our approach focusses on a combination of qualitative and quantitative goals. Moreover, we verify goal-reachability itself rather than separating goals and temporal logic properties of interest.

Nash equilibria with regard to quantitative aspects have been considered in [23–26]. In [23] the authors propose an extension of the temporal logic LTL called LTL[$\mathcal{F}$]. The evaluation of an LTL[$\mathcal{F}$] formula on a system model may yield a real value from the interval $[0, 1]$ rather then just a Boolean value. Hence, a formula may be *partially* satisfied by a system. It is shown that the Nash equilibrium synthesis problem for systems where agents have LTL[$\mathcal{F}$] goals is 2EXPTIME-complete. Contrary to our work, the LTL[$\mathcal{F}$] Nash equilibrium synthesis considers *purely quantitative* goals in the sense of maximising the satisfaction value of LTL[$\mathcal{F}$] formulas. In contrast, our approach combines qualitative goal formulas φ with quantitative pay-offs. The authors of [24,25] introduce multi-agent systems in which agents have a primary goal that is qualitative and a secondary goal that is quantitative. This is similar to our properties of interest where agents must achieve their

resource goal at least once (qualitative) and preferably as frequent resp. early as possible (quantitative). In [24] is proven that if the qualitative goals are Büchi conditions, then an *NP*-algorithm for solving the corresponding Nash equilibrium synthesis problem exists. Moreover, in [25] it is shown that if the qualitative goals are LTL formulas then the problem is 2EXPTIME-complete. The contributions of [24,25] are of theoretical nature, whereas our work also comprises tool support and experimental results. A further difference to our work is that the approaches proposed in [24,25] are limited to decide the existence of *strict Nash equilibria*, which form a subset of the (general) Nash equilibria that can be synthesised with our technique. Another framework for reasoning about systems where agents have both qualitative and quantitative goals is proposed in [26]. The authors augment multi-agent systems with transition pay-offs and introduce a quantitative extension of the logic ATL*. This allows to model check whether agents have the strategic abilities to achieve quantitative pay-off goals and at the same time qualitative state-based goals. The above works are predominantly of theoretical nature and focus on establishing general complexity results of the proposed synthesis problems. In contrast, our work considers a more specific scenario where agents primarily need to achieve qualitative resource goals with in a given time frame and secondarily prefer to reach the goals as early as possible or as often as possible. Moreover, our work includes a practical approach to solve this problem, which is based on a Boolean encoding and maximum satisfiability solving (Max-SAT) [7]. To the best of our knowledge, our technique is the first Max-SAT-based approach to the synthesis of optimal and Nash equilibrium strategies. In related fields, Max-SAT has been employed to find optimal coalitions of agents [27], to synthesise optimal controllers [28], and to model check quantitative hyper-properties [29]. [27] proposes a Max-SAT-based solution to the *coalition structure generation problem*, which is to find a partition of agents into coalitions such that the overall system pay-off gets maximised. Thus, in contrast to our approach the outcome of the technique presented in [27] is a *partition*, and not a strategy. The authors of [28] use Max-SAT to find a controller of an autonomous system that, given a set of LTL formulas, minimises the number of formulas that are violated by the system. The generated optimal controller is represented as a transition system. Thus, the technique can be regarded as a *system synthesis* approach rather than a strategy synthesis approach. In [29] Max-SAT is employed to solve the HyperLTL model checking problem, which is a generalisation of LTL model checking allowing for simultaneous quantification over multiple paths. The proposed technique is a *decision procedure* for the model checking problem, but it does not involve any synthesis.

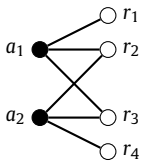
3. Multi-agent systems for resource allocation

In our approach we focus on model checking and synthesising strategies for *multi-agent systems for resource allocation* (MRAs), originally introduced in [1].

Definition 1 *(Multi-agent system for resource allocation).* A multi-agent system for resource allocation is a tuple $M = (Agt, Res, d, Acc)$ where

- $Agt = \{a_1, \ldots, a_n\}$ is a finite set of agents,
- $Res = \{r_1, \ldots, r_m\}$ is a finite set of resources,
- $d : Agt \to \mathbb{N}$ is a demand function that defines the number of resources that each agent needs to allocate in order to achieve its individual goal,
- $Acc : Agt \to 2^{Res}$ is an accessibility function that defines the subset of resources that each agent can access.

Example 1. The graph on the right describes the agents a_1, a_2, the resources r_1, r_2, r_3, r_4, and the accessibility function of the multi-agent system for resource allocation. The MRA is fully specified once the demand function is defined, e.g., $d(a_1) = 2, d(a_2) = 3$. In a sensor network context, this MRA may represent a heterogeneous WSN system consisting of an application for fire detection a_1 and an application for air quality monitoring a_2. The resources r_1 to r_4 may correspond to nodes with sensors for temperature, smoke, carbon dioxide, and ozone. Access to the sensors for smoke and carbon dioxide is shared, whereas the sensors for temperature and ozone are exclusive to a_1 resp. a_2. The demand function may model that application a_1 can confidently detect a fire based on data from two of the three sensors it can access, and that application a_2 requires data from all three sensors it can access in order to report on the air quality.



Each agent has the goal to gradually allocate a number of resources such that its demand is finally satisfied. The actions that can be performed for this are:

Definition 2 *(Actions).* Given an MRA M, the set of actions Act is the union of the following types of actions:

- request actions: $\{req_r^a \mid a \in Agt, r \in Acc(a)\}$
- release actions: $\{rel_r^a \mid a \in Agt, r \in Acc(a)\}$

- release-all actions: $\{rel^a_{all} \mid a \in Agt\}$
- idle actions: $\{idle^a \mid a \in Agt\}$

Hence, an agent can request a particular resource, release a particular resource that it currently holds, release all resources that it currently holds, or just idle. The fact that agents may hold certain resources gives rise to the notion of *states* of an MRA, which we subsequently define. An MRA runs in discrete rounds where in each round each agent chooses its next action. In a round the tuple of chosen actions, one per agent, gets executed simultaneously. The execution of actions leads to an evolution of the system between different states over time.

Definition 3 *(States).* A state of an MRA M is a function $s : Res \to Agt^+$ where $Agt^+ = Agt \cup \{a_0\}$ and a_0 is a dummy agent. If $s(r) = a_0$ then resource r is unallocated in state s. If $s(r) = a_i$ and $i > 0$ then r is allocated by agent a_i in s. We denote by s_0 the initial state of M, where $s(r) = a_0$ for each $r \in Res$, i.e., initially all resources are unallocated. We denote by S the set of all possible states of M. If we want to express that resource r is currently allocated by agent a_i but the current state is not further specified, then we simply write $r = a_i$.

Hence, states describe the current allocation of resources by agents. An agent may not be able to observe the entire state of the MRA. We assume that agents can only observe the (state of the) resources they have access to.

Definition 4 *(State observations).* Let M be an MRA, let $a_i \in Agt$ and let $s \in S$. Then the observation of agent a_i in state s is a function $s_{a_i} : Acc(a_i) \to Agt^+$ such that $s_{a_i}(r) = s(r)$ for all $r \in Acc(a_i)$. We denote by S_{a_i} the set of all possible state observations of a_i.

In our example, agent a_1 cannot access resource r_4. Hence, the state observation of this agent in a state where all resources are available and in a state where r_4 is the only allocated resource would be the same. In each state only a subset of actions may be available for execution by an agent, which we call the protocol:

Definition 5 *(Action availability protocol).* The action availability protocol is a function $P : S \times Agt \to 2^{Act}$ defined for each $s \in S$ and $a \in Agt$:

1. if $|s^{-1}(a)| = d(a)$ then $P(s, a) = \{rel^a_{all}\}$;
2. otherwise:
   (a) $rel^a_{all} \notin P(s, a)$;
   (b) $req^a_r \in P(s, a)$ iff $s(r) = a_0$;
   (c) $rel^a_r \in P(s, a)$ iff $s(r) = a$;
   (d) $idle^a \in P(s, a)$ iff $\forall r \in Acc(a) : s(r) \neq a_0$.

Thus, if an agent has reached its goal, it has to release all of its allocated resources. Otherwise, an agent can request an accessible resource that is currently unallocated, an agent can release a resource that it currently holds, and an agent can idle only if none of its accessible resources are currently available.

Definition 6 *(Action profiles).* An action profile in an MRA M is a mapping $ap : Agt \to Act$. AP denotes the set of all action profiles. We say that a profile ap is executable in a state $s \in S$ if for each $a \in Agt$ we have that $ap(a) \in P(s, a)$.

Based on action profiles we can formally define the evolution of an MRA.

Definition 7 *(Evolution).* The evolution of an MRA is a relation $\delta \subseteq S \times AP \times S$ where $(s, ap, s') \in \delta$ iff ap is executable in s and for each $r \in Res$:

1. if $s(r) = a_0$ then:
   (a) if $\exists a : ap(a) = req^a_r \wedge \forall a' \neq a : ap(a') \neq req^{a'}_r$ then $s'(r) = a$;
   (b) otherwise $s'(r) = a_0$;
2. if $s(r) = a$ for some $a \in Agt$ then:
   (a) if $ap(a) = rel^a_r \vee rel^a_{all}$ then $s'(r) = a_0$;
   (b) otherwise $s'(r) = a$.

If an action profile is executed in a state of an MRA M, this leads to a transition of M into a corresponding successor state, i.e., a change in the allocation of resources according to the actions chosen by the agents. According to the evolution, the request of a resource r by an agent a will be only successful if a is the only agent that requests r in the current round. If multiple agents request the same resource at the same time, then none of the agents will obtain it.

We are interested in solving strategic model checking problems with regard to MRAs: Given a coalition of agents $A \subseteq Agt$, does this coalition have a *uniform strategy* that guarantees that all agents in A will eventually achieve their goal, irrespective of how the opposition of agents $Agt \backslash A$ acts?

Definition 8 *(Uniform strategy).* A uniform strategy of an agent $a \in Agt$ in an MRA is an injective function $\alpha_a : S_a \to Act$. A strategy can be also denoted by a relation $\alpha_a \subseteq S_a \times Act$ where $\alpha_a(s_a, act^a) = true$ iff $\alpha_a(s_a) = act^a$. Given $A = \{a_1, \ldots, a_r\} \subseteq Agt$, a joint strategy for A is a tuple of strategies $\alpha_A = (\alpha_a)_{a \in A}$, one for each $a \in A$.

A strategy determines which action an agent will choose under which observation. A strategy is uniform if the following holds: Each time when an agent makes the same observation, it will perform the same action according to the strategy. If a coalition of agents follows a joint strategy, this can have *multiple* possible execution paths as outcomes because the remaining agents outside the coalition may act in an arbitrary way. In our approach, we assume that the remaining agents may follow an arbitrary strategy from a set Σ. The outcome of a strategy α_A in a state s for a set of opposition's strategies Σ is a set of paths.

Definition 9 *(Outcome of a strategy).* Let M be an MRA, s a state of M, $A \subseteq Agt$ and $B = Agt \backslash A$. Moreover, let α_A be a joint strategy for A and Σ a set of joint strategies for B. Then the outcome of α_A in state s, assuming that the agents in B follow an arbitrary strategy from Σ, is a set of paths

$$
\begin{aligned}
\Pi(s, \alpha_A, \Sigma) = \Big\{ \; & \pi = s_0 s_1 \ldots \mid s_0 = s \wedge \\
& \forall \beta_B \in \Sigma : \forall t \in \mathbb{N} : \exists (act_t^{a_1}, \ldots, act_t^{a_n}) \forall a \in Agt : \\
& \big(act_t^a \in P(s_t, a) \wedge (a \in A \to \alpha_a\big((s_t)_a\big) = act_t^a) \wedge \\
& (a \in B \to \beta_a\big((s_t)_a\big) = act_t^a) \wedge \\
& \big(s_t, (act_t^{a_1}, \ldots, .act_t^{a_n}), s_{t+1}\big) \in \delta\big) \Big\}
\end{aligned}
$$

where $(s_t)_a$ denotes the observation of agent a in state s_t.

The logic that we introduce for specifying strategic goal-achievability properties of agents in MRAs we call 1-ATL*. 1-ATL* is based on a subset of the alternating-time temporal logic ATL* [3].

Definition 10 *(1-ATL* syntax).* Let M be an MRA, $A \subseteq Agt$, $B = Agt \backslash A$ and Σ a set of joint strategies for B. Then formulas $\langle\langle A, \Sigma \rangle\rangle \varphi \in$ 1-ATL* over M are defined as follows:

$$\varphi := a.goal \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathbf{F}\varphi$$

where $a \in Agt$ and $a.goal$ is an atomic proposition that expresses that agent a has reached its goal, i.e., $s(a.goal) = true$ iff $|s^{-1}(a)| = d(a)$ for $s \in S$.

Here **F** refers to 'finally'. 1-ATL* formulas are restricted to a single strategic operator $\langle\langle A, \Sigma \rangle\rangle$ at the beginning of a formula. Since we follow a SAT-based bounded model checking approach [13] we define bounded semantics for 1-ATL*.

Definition 11 *(Bounded 1-ATL* semantics).* Let M be an MRA, let $s \in S$ be a state of M, let $k \in \mathbb{N}$. Moreover, let $A \subseteq Agt$, $a \in Agt$, $B = Agt \backslash A$ and Σ a set of joint strategies for B. Then the k-bounded evaluation of a 1-ATL* formula $\langle\langle A, \Sigma \rangle\rangle \varphi$ on the state s, written $[M, s \models_k \langle\langle A, \Sigma \rangle\rangle \varphi]$, is inductively defined as:

$$
\begin{aligned}
[M, s \models_k \langle\langle A, \Sigma \rangle\rangle \varphi] &\equiv \exists \alpha_A \forall \pi \in \Pi(s, \alpha_A, \Sigma) : [M, \pi \models_k \varphi] \\
[M, \pi \models_k a.goal] &\equiv |\pi(0)^{-1}(a)| = d(a) \\
[M, \pi \models_k \mathbf{F}\varphi] &\equiv \exists 0 \leq t \leq k : [M, \pi(t) \models_k \varphi]
\end{aligned}
$$

where $\pi(t)$ denotes the t-th state of the path π. Moreover, Boolean operators $\neg, \vee, \wedge$ are interpreted with the usual semantics.

3.1. Coalition versus opposition strategy synthesis

While ATL* uses strategic operators of the form $\langle\langle A \rangle\rangle$, we use extended strategic operators $\langle\langle A, \Sigma \rangle\rangle$. The semantic difference is as follows: A formula $\langle\langle A \rangle\rangle \varphi$ expresses that the coalition A has a universal strategy to achieve φ, irrespective of how the opposition $Agt \backslash A$ acts, whereas $\langle\langle A, \Sigma \rangle\rangle \varphi$ expresses that the coalition A has a strategy to achieve φ against all the

opposition's strategies in the set Σ. If we include all possible strategies of the opposition in Σ, then $\langle\langle A\rangle\rangle\varphi$ and $\langle\langle A, \Sigma\rangle\rangle\varphi$ are semantically identical. In our SAT-based approach we focus on solving strategic bounded model checking problems of the following form:

$$[M, s_0 \models_k \langle\langle A, \Sigma\rangle\rangle \Big(\bigwedge_{a\in A} \big(\mathbf{F} a.goal\big)\Big)]$$

Thus, we check whether the coalition A has a uniform strategy guaranteeing that each agent in A will finally reach its resource goal within at most k time steps, assuming that the opposition follows an arbitrary strategy in Σ. Our technique does not only yield the model checking result but also returns a winning strategy for A if such a strategy exists. We will also show how our approach can be used for solving the corresponding *universal* problem

$$[M, s_0 \models_k \langle\langle A\rangle\rangle \Big(\bigwedge_{a\in A} \big(\mathbf{F} a.goal\big)\Big)]$$

where the strategies of the opposition are not restricted by Σ. We call this problem the *coalition versus opposition* strategy synthesis.

4. SAT-encoding of coalition versus opposition strategy synthesis

We now present our propositional logic encoding of *coalition versus opposition* strategic bounded model checking problems $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle\varphi]$. We construct a propositional logic formula $[M, \langle\langle A, \Sigma\rangle\rangle\varphi, k]$ over a set of Boolean variables $Vars$ that is satisfiable if and only if the encoded model checking problem holds. We assume that the formula $[M, \langle\langle A, \Sigma\rangle\rangle\varphi, k]$ is converted into conjunctive normal form, which is the standard input format of SAT solvers:

Definition 12 *(Conjunctive normal form (CNF)).* Let Var be a set of Boolean variables. A propositional logic formula $\mathcal{F}$ over Var in *conjunctive normal form* is a conjunction of clauses $\mathcal{C}$ where each clause is a disjunction of literals l, and a literal is either a variable $v \in Var$ or its negation $\neg v$.

For CNF formulas the satisfiability problem is defined as follows:

Definition 13 *(Boolean satisfiability problem).* Let $\mathcal{F}$ over Var be a formula in conjunctive normal form. The Boolean satisfiability problem with regard to $\mathcal{F}$ is the problem of determining whether there exists a truth assignment $\alpha : Var \to \{\mathbf{0}, \mathbf{1}\}$ that makes all clauses of $\mathcal{F}$ true.

For CNF formulas $\mathcal{F}$ over a set of variables Var and assuming that $\mathcal{A}(Var)$ is the set of all possible truth assignments over Var we define Boolean satisfiability as the function

$$\mathbf{sat}(\mathcal{F}) = \begin{cases} \mathbf{1} & \text{if } \exists\alpha \in \mathcal{A}(Var) \text{ with } \alpha(\mathcal{F}) = \mathbf{1}, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

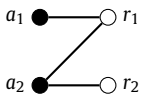
We will show that if the formula $[M, \langle\langle A, \Sigma\rangle\rangle\varphi, k]$ is satisfiable for a truth assignment $\alpha : Vars \to \{\mathbf{0}, \mathbf{1}\}$, then α describes a uniform strategy α_A for the coalition A that guarantees that the goal φ will be reached against all the opposition's strategies in Σ. Since we have the correspondence between truth assignments and strategies we denote both of them by α resp. α_A. In a top-down manner now we break down the overall encoding into sub encodings:

$$[M, \langle\langle A, \Sigma\rangle\rangle\varphi, k] = [\langle\langle A\rangle\rangle, k] \wedge \bigwedge_{\beta\in\Sigma} \big([\beta, k] \wedge [M, k]^\beta \wedge [\varphi, k]^\beta\big)$$

The sub formula $[\langle\langle A\rangle\rangle, k]$ encodes the condition that the agents in A must follow a uniform strategy and adhere to the protocol at each time step up to k. $[\beta, k]$ encodes that the agents in $B = Agt\backslash A$ exactly follow the strategy β. $[M, k]^\beta$ encodes all k-bounded paths of M starting in the initial state, and $[\varphi, k]^\beta$ is a constraint that restricts the paths of M to those that satisfy φ. Since for each strategy $\beta \in \Sigma$ different paths may be taken and sub formulas of φ may be satisfied in different states, we have for each β a distinct copy of the encoding of M and φ, indicated by the superscript. This means, for some $\beta \neq \beta'$ the encodings $[M, k]^\beta$ and $[M, k]^{\beta'}$ are structurally identical, but they are defined over distinct sets of variables. Henceforth, we typically omit the superscript, unless we want to emphasise that the encoding refers to a particular strategy β.

Besides defining the general propositional logic encoding, we also illustrate the encoding based on a running example that is split into parts 2(a) to 2(f):

Example 2(a). We consider simple example MRA that consists of two agents and two resources. Initially, both resources are unallocated. The accessibility of agents to resources is defined by the graph on the right. We assume that agent a_1 has a demand of 1 and agent a_2 has a demand of 2. Moreover, we assume a single-agent coalition $A = \{a_1\}$. Hence, the opposition is $B = \{a_2\}$. In our scenario, the set of strategies of the opposition is $\Sigma = \{\beta\}$ where β is informally defined as follows: If r_2 is available then request r_2, if r_2 is not available but r_1 is available then request r_1, if both resources are allocated by a_2 then release all, otherwise idle. The corresponding strategy synthesis problem is to determine whether there exists a strategy α for agent a_1 that guarantees that a_1 will achieve its goal of eventually holding one resource, under the assumption that agent a_2 follows the strategy β. It is easy to see that any strategy α where agent a_1 requests resource r_1 in the initial state is a winning strategy against β. In the further parts of the example we will illustrate how to construct the 1-bounded encoding $[M, \langle\langle\{a_1\}, \{\beta\}\rangle\rangle \mathbf{F} a_1.goal, 1]$ of this simple strategy synthesis problem. Moreover, we will show that a satisfying truth assignment of the encoding characterises a winning strategy for agent a_1.



4.1. Overall encoding

Subsequently, we present the details of the overall encoding. The encoding makes use of a number of basic encodings: $[r = a]_t$ denoting that resource r is allocated by agent a in the state at time step t, $[act^a]_t$ denoting that agent a chooses action act in the state at time step t, and $[a.goal]_t$ denoting that a has reached its goal in the state at step t. If the SAT solver generates a satisfying truth assignment α of the overall encoding and $\alpha([r = a]_t) = \mathbf{1}$ holds, then α characterises a path where at the t-th state resource r is allocated by agent a. Similar properties hold for the remaining basic encodings. For now we will remain with these informal definitions of these basic encodings. The formal definitions will follow in the next sub section. We start with the encoding of paths and the temporal logic formula before we consider the strategic parts $[\langle\langle A\rangle\rangle, k]$ and $[\beta, k]$. The encoding of k-bounded paths $[M, k]$ is composed of the following sub encodings

$$[M, k] = [Init]_0 \wedge \bigwedge_{t=0}^{k-1} [Evolution]_{t,t+1}$$

where $[Init]_0$ encodes the initial state at time step 0 and $[Evolution]_{t,t+1}$ encodes the evolution (Definition 7) of M from time step t to step $t + 1$.

Definition 14 *(Encoding of the initial state).* The encoding of the initial state of an MRA M at time step 0 where all resources are unallocated is

$$[Init]_0 = \bigwedge_{r \in Res} [r = a_0]_0$$

where $[r = a_0]_0$ is defined according to the encoding of resource states (Section 4.2).

Example 2(b). For our running example we get the initial state encoding $[Init]_0 = [r_1 = a_0]_0 \wedge [r_2 = a_0]_0$ which encodes that both r_1 and r_2 are unallocated at time step 0. A truth assignment α that satisfies the overall encoding must clearly also satisfy $[Init]_0$. Hence, for such an α we will have $\alpha([r_1 = a_0]_0) = \mathbf{1}$ and $\alpha([r_2 = a_0]_0) = \mathbf{1}$. Our encoding is defined in a way such that at each time step each resource is either unallocated or allocated by exactly one agent. Thus, from $\alpha([r_1 = a_0]_0) = \mathbf{1}$ it follows that $\alpha([r_1 = a_1]_0) = \mathbf{0}$ and $\alpha([r_1 = a_2]_0) = \mathbf{0}$, and we get an analogous property with regard to assignment α and resource r_2. In the subsequent part of the example we will see that this property of our encoding ensures that a truth assignment can be only satisfying if it characterises a state transition that is in line with the evolution (Definition 7).

The initial state at time step 0 is the only state that is fixed in our encoding. States that will be reached at time steps $1 \leq t \leq k$ follow from the truth assignment generated by the solver. The subsequent encoding ensures that generated sequences of states are in line with the evolution:

Definition 15 *(Encoding of the evolution).* The evolution of a multi-agent system for resource allocation M from time step t to $t + 1$ is encoded as $[Evolution]_{t,t+1} = \bigwedge_{r \in R} [r.evolution]_{t,t+1}$ where $[r.evolution]_{t,t+1} =$

$$
\begin{aligned}
\bigvee_{a\in Acc^{-1}(r)} \Big(& \big([r=a]_{t+1} \ \wedge\ [req_r^a]_t \wedge \textstyle\bigwedge_{a'\neq a} \neg[req_r^{a'}]_t\big) \\
& \vee \big([r=a]_{t+1} \ \wedge\ [r=a]_t \wedge \neg[rel_r^a]_t \wedge \neg[rel_{all}^a]_t\big) \\
& \vee \big([r=a_0]_{t+1} \ \wedge\ [rel_r^a]_t\big) \\
& \vee \big([r=a_0]_{t+1} \ \wedge\ [r=a]_t \wedge [rel_{all}^a]_t\big) \\
\Big) & \\
& \vee \big([r=a_0]_{t+1} \ \wedge\ [r=a_0]_t \wedge \textstyle\bigwedge_{a\in Acc^{-1}(r)} \neg[req_r^a]_t\big) \\
& \vee \big([r=a_0]_{t+1} \ \wedge\ [r=a_0]_t \wedge \textstyle\bigvee_{a,a'\in Acc^{-1}(r), a\neq a'} ([req_r^a]_t \wedge [req_r^{a'}]_t)\big)
\end{aligned}
$$

and the sub encodings are defined according to the encoding of resource states and actions (Section 4.2).

The encoding of the evolution has a sub formula for each resource r in M. It describes how the allocation state of r changes based on agent actions at particular time steps. The first line of the encoding expresses that in the state at the next time step $t+1$ the resource r will be allocated by agent a if at the current time step t agent a requests r and no other agent requests r.

Example 2(c). The following formula is the part of the evolution encoding of our example MRA focussing on resource r_1, agent a_1 and the evolution from time step 0 to time step 1:

$$
\begin{aligned}
& \big([r_1=a_1]_1 \ \wedge\ [req_{r_1}^{a_1}]_0 \wedge \neg[req_{r_1}^{a_2}]_0\big) \\
\vee\ & \big([r_1=a_1]_1 \ \wedge\ [r_1=a_1]_0 \wedge \neg[rel_{r_1}^{a_1}]_0 \wedge \neg[rel_{all}^{a_1}]_0\big) \\
\vee\ & \big([r_1=a_0]_1 \ \wedge\ [r_1=a_1]_0 \wedge [rel_{r_1}^{a_1}]_0\big) \\
\vee\ & \big([r_1=a_0]_1 \ \wedge\ [r_1=a_1]_0 \wedge [rel_{all}^{a_1}]_0\big) \\
\vee\ & \big([r_1=a_0]_1 \ \wedge\ [r_1=a_0]_0 \wedge \neg[req_{r_1}^{a_1}]_0 \wedge \neg[req_{r_1}^{a_2}]_0\big) \\
\vee\ & \big([r_1=a_0]_1 \ \wedge\ [r_1=a_0]_0 \wedge [req_{r_1}^{a_1}]_0 \wedge [req_{r_1}^{a_2}]_0\big)
\end{aligned}
$$

The first two lines of this formula encode the following: Resource r_1 will be allocated by agent a_1 at time step 1 (encoded as $[r_1=a_1]_1$) if a_1 is the only agent that requests r_1 at time step 0 (encoded as $[req_{r_1}^{a_1}]_0 \wedge \neg[req_{r_1}^{a_2}]_0$), or if a_1 already holds r_1 at time step 0 and does not release it (encoded as $[r_1=a_1]_0 \wedge \neg[rel_{r_1}^{a_1}]_0 \wedge \neg[rel_{all}^{a_1}]_0$). The remaining lines encode the conditions under which resource r_1 will be unallocated at time step 1. One condition is that agent a_1 currently holds r_1 and releases this resource at time step 0.
In the previous part of the running example, we already argued that the encoding of the initial state gives us constraints with regard to satisfying truth assignments α. In particular, $\alpha([r_1=a_0]_0) = \mathbf{1}$ must hold. This constraint also limits the evolution from time step 0 to 1. Cases where it is assumed that r_1 is allocated by a_1 at time step 0 become infeasible. Under the assumption of the above-mentioned partial assignment α, the evolution encoding can be reduced to the following feasible cases:

$$
\begin{aligned}
& \big([r_1=a_1]_1 \ \wedge\ [req_{r_1}^{a_1}]_0 \wedge \neg[req_{r_1}^{a_2}]_0\big) \\
\vee\ & \big([r_1=a_0]_1 \ \wedge\ [r_1=a_0]_0 \wedge \neg[req_{r_1}^{a_1}]_0 \wedge \neg[req_{r_1}^{a_2}]_0\big) \\
\vee\ & \big([r_1=a_0]_1 \ \wedge\ [r_1=a_0]_0 \wedge [req_{r_1}^{a_1}]_0 \wedge [req_{r_1}^{a_2}]_0\big)
\end{aligned}
$$

In our approach, such a reduction is performed by the SAT solver while searching a satisfying truth assignment of the overall encoding. For time step 0 to 1 it is quite obvious to see which evolution cases are feasible and which not, because this immediately follows from the initial state constraint $[Init]_0$. For later times steps the feasibility additionally depends on the actions that the agents took in the past. We will see that decisions of agents to take certain actions follow from the truth assignment generated for the overall encoding.

We are interested in goal-reachability properties: Do the agents in A have a strategy to achieve the goal φ within k steps? The strategic part of the property gets encoded separately. The temporal logic part gets encoded as follows:

Definition 16 *(Encoding of goal-reachability formulas).* Let M be an MRA, $A \subseteq Agt$ and $k \in \mathbb{N}$. Then the k-bounded goal-reachability property $\varphi = \bigwedge_{a\in A} \boldsymbol{F}(a.goal)$ is encoded in propositional logic as

$$
[\varphi, k] = \bigwedge_{a\in A} \Big(\bigvee_{t=0}^{k} [a.goal]_t \Big)
$$

where $[a.goal]_t$ is defined according to the encoding of goals (Section 4.2).

If we conjunct the encoding $[M, k]$ with $[\varphi, k]$, this restricts the k-bounded paths of M to those where each agent in A reaches its goal at least once. What we have encoded so far corresponds to a classical linear temporal logic bounded model checking problem for MRAs.

Example 2(d). In our running example the coalition consists of the single agent a_1 only. The corresponding goal reachability property in a 1-bounded scenario is:

$$[\varphi, 1] = [a_1.goal]_0 \vee [a_1.goal]_1 = [r_1 = a_1]_0 \vee [r_1 = a_1]_1$$

Thus, a satisfying truth assignment α must characterise a strategy where the agent is guaranteed to reach its goal at time step 0 or 1. Since the agent has only access to the resource r_1 and its demand is 1, the only possibility to achieve the goal is to successfully allocate r_1 within the bound. Clearly, holding resource r_1 at time step 0 contradicts the initial state constraint. Hence, for a winning strategy resp. satisfying truth assignment $\alpha([r_1 = a_1]_1) = \mathbf{1}$ must hold. In more general scenarios where agents have higher demands and access to multiple resources various possibilities to achieve a goal may exist.

We now extend the encoding with the strategic aspects of 1-ATL*. The strategic encodings uses two additional basic encodings: $[uniform.act^a]_t$ denoting that agent a chooses action act in the state at time step t and also at all other time steps where the same state observation is present, and $[s_a]_t$ denoting that the observation of agent a at time step t is s_a.

Definition 17 *(Encoding of the protocol).* Let M be an MRA, let $A \subseteq Agt$ and let $k \in \mathbb{N}$. Then the protocol of A for all time steps up to k is encoded in propositional logic as $[\langle\langle A\rangle\rangle, k] = \bigwedge_{t=0}^{k} \bigwedge_{a \in A} [a.protocol]_t$ where $[a.protocol]_t =$

$$\begin{aligned}
\bigvee_{r \in Acc(a)} \Big(& \big([uniform.req_r^a]_t \ \wedge\ \neg[a.goal]_t \wedge [r = a_0]_t\big) \\
& \vee \big([uniform.rel_r^a]_t \ \wedge\ \neg[a.goal]_t \wedge [r = a]_t\big) \\
\Big) & \\
\vee\ & \big([uniform.rel_{all}^a]_t \ \wedge\ \ [a.goal]_t\big) \\
\vee\ & \big([uniform.idle^a]_t \ \wedge\ \neg[a.goal]_t \wedge \textstyle\bigwedge_{r \in Acc(a)} \neg[r = a_0]_t\big)
\end{aligned}$$

and the sub encodings are defined according to the encoding of actions with uniformity constraints, goals, and resource states (Section 4.2).

The constraint $[\langle\langle A\rangle\rangle, k]$ forces the agents in A to follow the protocol at all time steps up to k. This means only actions that are available in the current state can be chosen. Moreover, the constraint enforces the uniformity of choices with regard to the state observation. The first line of the protocol encoding ensures that some agent a can only request some resource r if the agent has not reached its goal yet and r is unallocated in the state at the current time step. Furthermore, the sub constraint $[uniform.req_r^a]_t$ ensures that if the agent chooses the req_r^a action in the state at time step t, then it has to choose the same action at all time steps where the agent's state observation is the same as at t.

Example 2(e). The following formula is the part of the protocol encoding of our example MRA focussing on agent a_1 and its strategic decision at time step 0:

$$\begin{aligned}
& \big([uniform.req_{r_1}^{a_1}]_0 \ \wedge\ \neg[a_1.goal]_0 \wedge [r_1 = a_0]_0\big) \\
\vee\ & \big([uniform.rel_{r_1}^{a_1}]_0 \ \wedge\ \neg[a_1.goal]_0 \wedge [r_1 = a_1]_0\big) \\
\vee\ & \big([uniform.rel_{all}^{a_1}]_0 \ \wedge\ \ [a_1.goal]_0\big) \\
\vee\ & \big([uniform.idle^{a_1}]_0 \ \wedge\ \neg[a_1.goal]_0 \wedge \neg[r_1 = a_0]_0\big)
\end{aligned}$$

Thus, the agent a_1 can request the accessible resource r_1 (encoded as $[uniform.req_{r_1}^{a_1}]_0$) if the goal is not yet reached and the resource is currently available (encoded as $\neg[a_1.goal]_0 \wedge [r_1 = a_0]_0$). a_1 can release the resource if the goal is not yet reached and the agent currently holds the resource. The agent can release all resources if the goal is reached, and it can idle if the goal is not yet reached and the resource is currently unavailable. Since our example considers time step 0, the initial state constraint restricts the actions that the agent can actually choose. We have that in the initial state agent a_1 has not achieved its goal and that resource r_1 is unallocated. Under the assumption of a partial truth assignment that satisfies the initial state constraint, the protocol encoding can be reduced to the following feasible choice:

$$\big([uniform.req_{r_1}^{a_1}]_0 \ \wedge\ \neg[a_1.goal]_0 \wedge [r_1 = a_0]_0\big)$$

Consequently, the only action that agent a_1 can choose at time step 0 is to request resource r_1.

The final part of the encoding concerns strategies. In our approach, we use this part to fix the strategy that the opposition of agents $B = Agt \backslash A$ is following.

Definition 18 *(Encoding of strategies).* Let $A = \{a_1, \ldots, a_r\} \subseteq Agt$, let $\alpha_A = (\alpha_a)_{a \in A}$ be a joint strategy for A and let $k \in \mathbb{N}$. Then the prescription of the strategy α_A to A at all time steps up to k is encoded as

$$[\alpha_A, k] = \bigwedge_{t=0}^{k} \bigwedge_{a \in A} \bigwedge_{(s_a, act^a) \in \alpha_a} \left([s_a]_t \rightarrow [act^a]_t\right)$$

where $[s_a]_t$ is defined according to the encoding of state observations and $[act^a]_t$ is defined according to the encoding of actions (Section 4.2).

Each clause $\left([s_a]_t \rightarrow [act^a]_t\right)$ in this encoding ensures that if at some time step t the state observation s_a holds, then the agent a has to choose action act^a according to the strategy α_a.

Example 2(f). In our running example we assume that the opposing agent a_2 follows the strategy β, which we defined in part (a) of the example. The corresponding propositional logic encoding $[\beta, 0]$ looks as follows:

$$\begin{aligned}
&\left([r_2 = a_0]_0 \rightarrow [req^{a_2}_{r2}]_0\right) \\
\wedge\ &\left((\neg[r_2 = a_0]_0 \wedge [r_1 = a_0]_0) \rightarrow [req^{a_2}_{r1}]_0\right) \\
\wedge\ &\left(([r_1 = a_2]_0 \wedge [r_2 = a_2]_0) \rightarrow [rel^{a_2}_{all}]_0\right) \\
\wedge\ &\left(([r_1 = a_1]_0 \wedge [r_2 = a_2]_0) \rightarrow [idle^{a_2}]_0\right)
\end{aligned}$$

The first line of the above formula encodes the strategic decision to request resource r_2 (encoded as $[req^{a_2}_{r2}]_0$) if this resource is available in the state at time step 0 (encoded as $[r_2 = a_0]_0$). The remaining lines encode strategic decisions prescribed by β for the further possible states of the system. We know that at time step 0 the system is in the initial state where r_2 is available. Hence, any satisfying truth assignment α must satisfy $[r_2 = a_0]_0$ and consequently according to the above encoding also $[req^{a_2}_{r2}]_0$.

This completes the definition of the overall encoding $[M, \langle\langle A, \Sigma\rangle\rangle \varphi, k] = [\langle\langle A\rangle\rangle, k] \wedge \bigwedge_{\beta \in \Sigma} \left([\beta, k] \wedge [M, k]^\beta \wedge [\varphi, k]^\beta\right)$. For our running example we can obtain a concrete overall encoding $[M, \langle\langle\{a_1\}, \{\beta\}\rangle\rangle \mathbf{F} a_1.goal, 1] = [\langle\langle\{a_1\}\rangle\rangle, 1] \wedge [\beta, 1] \wedge [M, 1] \wedge [\mathbf{F} a_1.goal, 1]$ where the sub formulas are defined as outlined in the parts (a) to (f). As illustrated, this concrete encoding will be only satisfied for truth assignments that characterise the following: Initially all resources are unallocated, both agents adhere to the protocol, state transitions are conform with the evolution, agent a_2 follows the fixed strategy β, and agent a_1 has a strategy to achieve its goal within 1 time step. Thus, we exemplified how our SAT-based strategy synthesis approach conceptually works.

We continue with the formal definition of the basic encodings that are used within the overall encoding. This will be followed by a discussion of the properties of the overall encoding with regard to satisfiability results and derivable model checking resp. strategy synthesis results.

4.2. Basic encodings

An essential basic encoding in our approach is that a particular resource is allocated by a particular agent in the state at time step t. In the encoding we make use of the fact that the agents in an MRA are indexed from 0 to n where the 0-index indicates the dummy agent a_0 holding unallocated resources. Each index can be represented by an m-digit binary number, and each binary number can be logically represented by a conjunction of m negated or non-negated Boolean variables. We introduce m Boolean variables for each resource $r_j \in Res$ and encode that r_j is allocated by some agent $a_i \in Agt$ by building a conjunction that corresponds to the binary representation of the agent's index i:

Definition 19 *(Encoding of resource states).* Let M be multi-agent system for resource allocation, let $r_j \in Res$, let $a_i \in Agt^+$ and let $t \in \mathbb{N}$. Let $m = \lceil \log_2 |Agt^+| \rceil$ and let $b_{m-1} \ldots b_0$ be the m-digit binary representation of the agent's index number i. Then the allocation of resource r_j by agent a_i in the state at time step t is encoded as

$$[r_j = a_i]_t := \bigwedge_{l=m-1}^{0} \left(\left(b_l \wedge [r_j]^l_t\right) \vee \left(\neg b_l \wedge \neg [r_j]^l_t\right)\right)$$

where $[r_j]^l_t$ with $0 \leq l < m$ are the Boolean variables introduced for the encoding.

Note that in this encoding each b_l is a Boolean value (**0** or **1**), which means that either the left-hand side or the right-hand side of the disjunction evaluates to **0**. Thus, the encoding can be immediately simplified to a pure conjunction. Since we get a pure conjunction, it is excluded that at some step t a resource is falsely allocated by multiple agents: For some r and $a \neq a'$ there exists no truth assignment α such that $\alpha\left([r = a]_t\right) = \mathbf{1}$ and $\alpha\left([r = a']_t\right) = \mathbf{1}$. The conjunction over digits

is built from the left-most position $m-1$ to the right-most position 0. By following the definition above, we can encode a state where some resource r_1 is unallocated, r_2 is allocated by some agent a_1, and r_3 is allocated by a_2:

$$\left(\neg[r_1]_t^1 \wedge \neg[r_1]_t^0\right) \wedge \left(\neg[r_2]_t^1 \wedge [r_2]_t^0\right) \wedge \left([r_3]_t^1 \wedge \neg[r_3]_t^0\right)$$

Since $Agt^+ = \{a_0, a_1, a_2\}$, we introduce two Boolean variables per resource to be able to encode the binary representations **00**, **01**, **10**. Based on the encoding of resource states, we can now also encode state observations and goals of agents.

Definition 20 *(Encoding of state observations).* Let M be an MRA, $a \in Agt$, $s_a \in S_a$ and $t \in \mathbb{N}$. Then the observation s_a by a at step t is encoded as

$$[s_a]_t := \bigwedge_{r_j \in Acc(a)} [r_j = s_a(r_j)]_t$$

where $[r_j = s_a(r_j)]_t$ is defined according to the encoding of resource states.

Hence, the encoding of the state observation s_a by agent a at time step t is a conjunction over the states of accessible resources which are in line with s_a.

Definition 21 *(Encoding of goals).* Let M be an MRA, let $a \in Agt$ and let $t \in \mathbb{N}$. Then the achievement of a's goal in the state at time step t is encoded as

$$[a.goal]_t = \bigvee_{\substack{R \subseteq Acc(a) \\ |R| = d(a)}} \left(\bigwedge_{r \in R} [r = a]_t\right)$$

where $[r = a]_t$ is defined according to the encoding of resource states.

An agent a has achieved its goal at time step t if the number of resources allocated by a in the current state is equal to the demand $d(a)$ of this agent. Since the number of accessible resources may be higher than the demand, all possibilities for satisfying the demand need to be considered. In the example below, we assume that a has access to the resources r_1, r_2, r_3 and its demand is 2. As a corresponding goal encoding for time step t we get:

$$[a.goal]_t = \left([r_1 = a]_t \wedge [r_2 = a]_t\right) \vee \left([r_1 = a]_t \wedge [r_3 = a]_t\right) \vee \left([r_2 = a]_t \wedge [r_3 = a]_t\right)$$

If the solver generates an assignment α with $\alpha\left([a.goal]_t\right) = \mathbf{1}$, then on the path corresponding to α agent a has reach its goal in the state at time step t.

In the encoding of actions by agents we follow a similar concept as in the encoding of resource states. We assign a unique binary number to each possible action of an agent a_i and we represent each action by a logical conjunction over negated or non-negated Boolean variables associated with this agent.

Definition 22 *(Encoding of actions).* Let $a_i \in Agt$, $act^{a_i} \in Act(a_i)$, $t \in \mathbb{N}$ and $m = \lceil \log_2 |Act(a_i)| \rceil$. Moreover, let $f_{a_i} : Act(a_i) \to \{0, \ldots, m-1\}$ a bijection that assigns a unique number to each possible action of a_i and let $b_{m-1} \ldots b_0$ be the m-digit binary representation of $f_{a_i}(act)$. Then the action act^{a_i} of a_i at step t is encoded as

$$[act^{a_i}]_t := \bigwedge_{l=0}^{m-1} \left(b_l \wedge [ac_i]_t^l\right) \vee \left(\neg b_l \wedge \neg[ac_i]_t^l\right)$$

where $[ac_i]_t^l$ with $0 \leq l < m$ are the Boolean variables introduced for the encoding.

Assuming that some agent a_1 can perform 6 different actions, we need 3 Boolean variables for their encoding. Moreover, assuming that the number 0 is assigned to the action $idle^{a_1}$, the corresponding encoding is:

$$[idle^{a_1}]_t = \neg[ac_1]_t^2 \wedge \neg[ac_1]_t^1 \wedge \neg[ac_1]_t^0$$

In the remainder of this sub section, we show how we enforce uniform behaviour of the agents in A and how we include a logical mechanism that allows us to synthesise uniform strategies for reaching the goal. Since a strategy links a state observation s_a with an action act^a, we define strategic decision encodings $[s_a.act^a]$. We include these decision encodings in our overall encoding such that a truth assignment α satisfies $[s_a.act^a]$ if and only if the strategy characterised by α links s_a with act^a. Since strategic decisions are universal and not restricted to a particular time step, this encoding does not include t as a parameter.

Definition 23 *(Encoding of strategic decisions).* Let M be an MRA, let $a_i \in Agt$, let $act^{a_i} \in Act(a_i)$ and let $s_{a_i} \in S_{a_i}$. Moreover, let $m = \lceil \log_2 |Act(a_i)| \rceil$ and let $f_{a_i} : Act(a_i) \to \{0, \ldots, m-1\}$ be a bijection that assigns a unique number to each possible action of agent a_i. Let $b_{m-1} \ldots b_0$ be the m-digit binary representation of $f_{a_i}(act^{a_i})$. Then the strategic decision of agent a_i to perform action act^{a_i} in state observation s_{a_i} is encoded as

$$[s_{a_i}.act^{a_i}] := \bigwedge_{l=0}^{m-1} \left(b_l \wedge [sac_i]^l\right) \vee \left(\neg b_l \wedge \neg [sac_i]^l\right)$$

where $[sac_i]^l$ with $0 \leq l < m$ are the Boolean variables for the encoding.

Similarly to the encoding of actions, we assign a unique binary number to each possible action of an agent a_i and we represent each strategic decision by a logical conjunction over negated or non-negated Boolean variables associated with this agent. This ensures that for different actions $act^{a_i} \neq act'^{a_i}$ no truth assignment can satisfy $[s_{a_i}.act^{a_i}]$ and $[s_{a_i}.act'^{a_i}]$ at the same time. This results in a guaranteed uniformity of strategies synthesised from the decision encodings.

The final part of the basic encoding concerns the uniform choice of actions:

Definition 24 *(Encoding of actions with uniformity constraints).* Let $a_i \in Agt$, let $act^{a_i} \in Act(a_i)$ and let $t \in \mathbb{N}$. Then the uniformity constraint with regard to action act^{a_i} by agent a_i at time step t is encoded as

$$[uniform.act^{a_i}]_t := [act^{a_i}]_t \wedge \Big(\bigvee_{s_{a_i} \in S_{a_i}, act^{a_i} \in P(s_{a_i})} \left([s_{a_i}]_t \wedge [s_{a_i}.act^{a_i}]\right)\Big)$$

where $[s_{a_i}]_t$ and $[s_{a_i}.act^{a_i}]$ are defined according to the encoding of state observations and strategic decisions.

Here the uniform choice of actions by agents is enforced as follows: At each step t when an agent a makes a choice to perform action act^a, we connect this choice with the strategic decision encoding $[s_a.act^a]$ corresponding to the current state observation s_a and to act^a. This ensures that the action can only be chosen if there has been no time step with the same observation where a different action has been chosen, or synonymously, the same action is also chosen at all steps where the observation is the same as at t. This completes our encoding of strategic bounded model checking problems $[M, s \models_k \langle\langle A, \Sigma \rangle\rangle \varphi]$ into a propositional formula $[M, \langle\langle A, \Sigma \rangle\rangle \varphi, k]$. Next, we summarise the properties of the encoding.

4.3. Properties of the encoding

The major property of our encoding is that it allows to perform sound model checking of the encoded problem via satisfiability solving.

Theorem 1 *(Model checking). Let $[M, s \models_k \langle\langle A, \Sigma \rangle\rangle \varphi]$ be a strategic bounded model checking problem and let $[M, \langle\langle A, \Sigma \rangle\rangle \varphi, k]$ be its encoding over Vars. Then:*

$$[M, s \models_k \langle\langle A, \Sigma \rangle\rangle \varphi] \equiv \boldsymbol{sat}\big([M, \langle\langle A, \Sigma \rangle\rangle \varphi, k]\big)$$

Hence, the coalition A has a uniform strategy to achieve the goal φ within k time steps against all opposition's strategies in Σ if and only if the propositional logic encoding is satisfiable. Moreover, our approach also allows us to synthesise such a uniform strategy that guarantees the achievement of the goal φ:

Theorem 2 *(Strategy synthesis). Let $[M, s \models_k \langle\langle A, \Sigma \rangle\rangle \varphi]$ be a strategic bounded model checking problem, let $[M, \langle\langle A, \Sigma \rangle\rangle \varphi, k]$ be its encoding over Vars and let $\alpha : Vars \to \{\boldsymbol{0}, \boldsymbol{1}\}$ with $\alpha\big([M, \langle\langle A, \Sigma \rangle\rangle \varphi, k]\big) = \boldsymbol{1}$. Then for the strategy*

$$\alpha_A = \Big(\{(s_a, act^a) | s_a \in S_a \wedge act^a \in Act \wedge \alpha([s_a.act^a]) = \boldsymbol{1}\}_{a \in A}\Big)$$

the following holds: $\forall \pi \in \Pi(s, \alpha_A, \Sigma) : [M, \pi \models_k \varphi]$.

Thus, from a truth assignment α that satisfies the encoding we can directly derive a corresponding uniform strategy α_A that guarantees φ.

Proof of Theorem 1 and Theorem 2. The correctness of Theorem 1 and Theorem 2 is closely linked. The correctness follows from the subsequent lemmas.

Firstly, we show that the part $[M, k]$ of the overall encoding characterises k-bounded paths of M that are in line with the evolution.

Lemma 1 *(Evolution paths). Let* $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi]$ *be a strategic bounded model checking problem and let* $[M,k] = [Init]_0 \wedge \bigwedge_{t=0}^{k-1}[Evolution]_{t,t+1}$ *be the encoding of all k-bounded paths of M over Vars. Then for each truth assignment* $\alpha : Vars \to \{\mathbf{0},\mathbf{1}\}$ *with* $\alpha([M,k]) = \mathbf{1}$ *there exists a sequence of states* $\pi = s_0 \dots s_k$ *and a sequence of action profiles* $(act_t^{a_1}, \dots, act_t^{a_n}), 0 \le t < k$ *in M such that*

$$\forall 0 \le t \le k : \forall a \in Agt^+ : \forall r \in Res : s_t(r) = a \text{ iff } \alpha([r = a]_t) = \mathbf{1}$$

and

$$\forall 0 \le t < k : \delta(s_t, (act_t^{a_1}, \dots, act_t^{a_n}), s_{t+1}) = \mathbf{1} \text{ iff } \forall a \in Agt : \alpha([act^{a_i}]_t) = \mathbf{1}$$

Proof of Lemma 1. We have that s_0 is the initial state of M, where $s(r) = a_0$ for each $r \in Res$, i.e., initially all resources are unallocated. According to Definition 12, the encoding of the initial state is $[Init]_0 = \bigwedge_{r\in Res}[r = a_0]_0$. Hence, any truth assignment α that satisfies $[M,k]$ must have the property that $\alpha([r = a_0]_0) = \mathbf{1}$ for each $r \in Res$. Moreover, we have that the evolution of an MRA is a relation $\delta \subseteq S \times AP \times S$ where $(s, ap, s') \in \delta$ iff ap is executable in s and for each $r \in Res$ the conditions of Definition 7 hold. Further, we have that evolution of an MRA M from time step t to $t+1$ is encoded as $[Evolution]_{t,t+1} = \bigwedge_{r\in R}[r.evolution]_{t,t+1}$ (see Definition 13). Consequently, we get that $[Init]_0 \wedge [Evolution]_{0,1}$ only evaluates to $\mathbf{1}$ for assignments α such that for all $r \in Res$ $\alpha([r = a]_1) = \mathbf{1}$ if and only if there is a prefix $s_0 s_1$ in M and $s_1(r) = a$. Moreover, if according to the evolution, the agents $a \in Agt$ have chosen the actions act^a in state s_0, then $\alpha([act^a]) = \mathbf{1}$ must hold exactly for these actions. This argumentation can be extended to all prefixes $s_0 \dots s_k$ of length k, which completes the proof of Lemma 1. □

We now consider Lemma 2 which shows that there is an exact correspondence between k-prefixes of M for which the goal-achievability property φ holds and satisfying assignments of $[M,k] \wedge [\varphi,k]$.

Lemma 2 *(Evolution paths satisfying* φ*). Let* $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi]$ *be a strategic bounded model checking problem and let* $[M,k] \wedge [\varphi,k]$ *be the encoding of all k-bounded paths of M over Vars that satisfy* φ*. Then for each truth assignment* $\alpha : Vars \to \{\mathbf{0},\mathbf{1}\}$ *with* $\alpha([M,k]) = \mathbf{1}$ *there exists a sequence of states* $\pi = s_0 \dots s_k$ *and a sequence of action profiles* $(act_t^{a_1}, \dots, act_t^{a_n}), 0 \le t < k$ *in M such that all properties of Lemma 1 hold and additionally* $\forall a \in A : \exists 0 \le t \le k : \alpha([a.goal]_t) = \mathbf{1}$.

Proof of Lemma 2. The goal-achievability property is $\varphi = \left(\bigwedge_{a\in A}(\mathbf{F}a.goal)\right)$ where $\mathbf{F}a.goal$ holds for a k-bounded path π if $\exists 0 \le t \le k : |\pi(t)^{-1}(a)| = d(a)$ (Definition 10 an 11). The corresponding k-bounded encoding is $[\varphi,k] = \bigwedge_{a\in A}\left(\bigvee_{t=0}^{k}[a.goal]_t\right)$ where $[a.goal]_t = \bigvee_{R\subseteq Acc(a),|R|=d(a)}\left(\bigwedge_{r\in R}[r = a]_t\right)$ (Definition 16 and 21). If in some state $\pi(t)$ some agent a has reached its demand, then there must be some subset $R \subseteq Acc(a)$ such that the size of R equals the agent's demand and a holds all resources of R in state $\pi(t)$. So if there is a path $s_0 \dots s_k$ in M that satisfies $\left(\bigwedge_{a\in A}(\mathbf{F}a.goal)\right)$, then the truth assignment α corresponding to $s_0 \dots s_k$ must also have the following property: $\alpha(\bigwedge_{a\in A}\left(\bigvee_{t=0}^{k}[a.goal]_t\right)) = \mathbf{1}$. This completes the proof of Lemma 2. □

We now consider Lemma 3 which shows that there is an exact correspondence between k-prefixes of M for which the goal-achievability property φ holds and where the opposition B follows the fixed strategy β, and satisfying assignments of $[\beta,k] \wedge [M,k] \wedge [\varphi,k]$.

Lemma 3 *(Evolution paths satisfying* φ *and* β*). Let* $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi]$ *be a strategic bounded model checking problem and let* $[\beta,k] \wedge [M,k] \wedge [\varphi,k]$ *be the encoding of all k-bounded paths of M over Vars that satisfy* φ *and where the opposition B adheres to the strategy* β*. Then for each truth assignment* $\alpha : Vars \to \{\mathbf{0},\mathbf{1}\}$ *with* $\alpha([M,k]) = \mathbf{1}$ *there exists a sequence of states* $\pi = s_0 \dots s_k$ *and a sequence of action profiles* $(act_t^{a_1}, \dots, act_t^{a_n}), 0 \le t < k$ *in M such that all properties of Lemma 1 and Lemma 2 hold and additionally*

$$\forall b \in B : \forall act^b \in Act : \forall 0 \le t \le k : \beta((s_t)_b) = act^b \text{ iff } \alpha([s_b]_t \to [act^b]_t) = \mathbf{1}$$

where $(s_t)_b$ *denotes the observation of agent b in state* s_t.

Proof of Lemma 3. From the encoding of strategies (Definition 18) we get the following: Let $B = \{b_1, \dots, b_r\} \subseteq Agt$, let $\beta(\beta_{b_1}, \dots, \beta_{b_r})$ be a joint strategy for B and let $k \in \mathbb{N}$. Then the prescription of the strategy β to B at all time steps up to k is encoded as $[\beta,k] = \bigwedge_{t=0}^{k}\bigwedge_{b\in B}\bigwedge_{(s_b,act^b)\in\beta_b}\left([s_b]_t \to [act^b]_t\right)$ Hence, for a truth assignment α with $\alpha([\beta,k]) = \mathbf{1}$, we get that also $\alpha([s_b]_t \to [act^b]_t) = \mathbf{1}$ holds for each time step, each agent $b \in B$ and each $(s_b, act^b) \in \beta_b$. Thus, we have for all time steps along the path characterised by α that if the current state observation of some agent b is s_b, then the agent will perform action act^b, which means the agent adheres to the strategy β in all states of the path characterised by α. This completes the proof of Lemma 3. □

We now consider Lemma 4 which states that if we have the encoding of a strategic bounded model checking problem with $\Sigma = \{\beta\}$ where β is some strategy of the opposition, then only if there exists a satisfying assignment α there exists a uniform succeeding strategy for A and this strategy can be derived from α.

Lemma 4 *(Evolution paths satisfying φ, β and uniform protocol). Let $[M, s \models_k \langle\langle A, \{\beta\}\rangle\rangle \varphi]$ be a strategic bounded model checking problem and let $[\langle\langle A\rangle\rangle, k] \wedge [\beta, k] \wedge [M, k] \wedge [\varphi, k]$ be the encoding of all k-bounded paths of M over Vars that satisfy φ where the opposition B adheres to the strategy β and A follows the protocol in a uniform manner. Then for each truth assignment $\alpha : Vars \rightarrow \{\mathbf{0}, \mathbf{1}\}$ with $\alpha([M, k]) = \mathbf{1}$ there exists a sequence of states $\pi = s_0 \dots s_k$ and a sequence of action profiles $(act_t^{a_1}, \dots, act_t^{a_n}), 0 \leq t < k$ in M such that all properties of Lemma 1, 2 and 3 hold and additionally for the strategy $\alpha_A = \Big(\{(s_a, act^a) | s_a \in S_a \wedge act^a \in Act \wedge \alpha([s_a.act^a]) = \mathbf{1}\Big)$ the following holds: $\forall \pi \in \Pi(s, \alpha_A, \{\beta\}) : [M, \pi \models_k \varphi]$.*

Proof of Lemma 4. The protocol encoding is as follows: Let M be an MRA, let $A \subseteq Agt$ and let $k \in \mathbb{N}$. Then the protocol of A for all time steps up to k is encoded in propositional logic as $[\langle\langle A\rangle\rangle, k] = \bigwedge_{t=0}^{k} \bigwedge_{a \in A} [a.protocol]_t$ where $[a.protocol]_t$ is defined according to Definition 17. Hence, if $\alpha([\langle\langle A\rangle\rangle, k]) = \mathbf{1}$, then for each $a \in A$ and each t there exists some action act^a such that $\alpha([uniform.act^a]_t) = \mathbf{1}$ holds. Let $(s_a)_t$ be the state observation of agent a in the state at time step t along the path characterised by α. Then according to the encoding of actions with uniformity constraints (Definition 24) we get $\alpha([s_a.act^a]) = \mathbf{1}$. We can synthesise the strategy α_A as outlined in Lemma 4. Since the truth assignment α also satisfies all properties of the Lemmas 1, 2 and 3, we can conclude that the synthesised strategy α_A that the agents in A follow along the path characterised by α is a winning strategy for the goal-reachability property against the opposition's strategy β. This completes the proof on Lemma 4. □

Lemma 4 can be straightforwardly generalised to Theorem 2. Instead of synthesising a strategy α_A that succeeds against a single opposition's strategy β, we can also synthesise a strategy α_A that succeeds against all opposition's strategies in a set Σ. For this the encoding gets extended to $[\langle\langle A\rangle\rangle, k] \wedge \bigwedge_{\beta \in \Sigma} \left([\beta, k] \wedge [M, k]^\beta \wedge [\varphi, k]^\beta\right)$. A truth assignment α that satisfies this encoding characterises a strategy that is successful against all $\beta \in \Sigma$. We can conclude that Theorem 2 holds: Exactly the satisfying truth assignments α of $[M, \langle\langle A, \Sigma\rangle\rangle \varphi, k]$ characterise strategies α_A such that $\forall \pi \in \Pi(s, \alpha_A, \Sigma) : [M, \pi \models_k \varphi]$. holds. This immediately implies that $[\langle\langle A\rangle\rangle, k] \wedge \bigwedge_{\beta \in \Sigma} \left([\beta, k] \wedge [M, k]^\beta \wedge [\varphi, k]^\beta\right)$ is a correct encoding of $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi]$ in the sense that $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi] \equiv \mathbf{sat}\big([M, \langle\langle A, \Sigma\rangle\rangle \varphi, k]\big)$ holds. Hence, both Theorem 1 and Theorem 2 hold. □

5. Coalition versus opposition strategy synthesis algorithm

Our SAT-based approach allows to solve coalition versus opposition model checking problems of the form $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi]$ where the coalition A attempts to reach its goal against the opposition's strategies in Σ. However, it is typically of interest to synthesise a strategy that *universally* succeeds, i.e., against *all* possible strategies of the opposition. The common notation for this is: $[M, s \models_k \langle\langle A\rangle\rangle \varphi]$. Universal goal-achievability can be naively checked by including all possible strategies in Σ. But this would involve an exorbitant increase of the size of the encoding. We approach this problem by defining the iterative Algorithm 1 on the subsequent page that successively extends the strategy set Σ. In each iteration, two strategic model checking problems are solved: We first check whether $[M, s \models_k \langle\langle A, \Sigma\rangle\rangle \varphi]$ holds, i.e., whether A has a strategy α that succeeds against all strategies in Σ. If not, then we can immediately terminate with the result that the model checking problem does not hold. Otherwise, our algorithm will synthesise a strategy α that succeeds against Σ. Secondly, we consider the so-called complementary model checking problem $[M, s \models_k \langle\langle B, \{\alpha\}\rangle\rangle \neg\varphi]$, i.e., we check whether the opposition B has a strategy β that succeeds against α in preventing the coalition A from reaching the goal φ. If the opposition does not have such a strategy β, then we can conclude that α is a universally winning strategy for φ and the algorithm terminates with this result. Otherwise, we synthesise β, add it to Σ and run the next iteration. In our algorithm, we initialise Σ with a simple greedy strategy for B: As long as its goal is not reached and accessible resources are available, each agent in B requests the accessible and available resource r_j with the smallest index j.

The termination of Algorithm 1 is guaranteed. In each non-terminating iteration a strategy β is added to the set Σ that was not contained Σ before. Since MRAs have finitely many states and actions, the number of possible strategies is also finite. Thus, in the worst case the number of iterations is equal to the number of possible strategies for B. However, the concept of checking the original problem *and* the complementary problem in each iteration allows for early termination in many cases: If the complementary problem does not hold, then we already know that the current α is a universally winning strategy – even if we have not considered all possible strategies for B yet. Since our approach is based on *bounded* model checking it is incomplete, i.e., only bounded goal-reachability can be checked. However, if we can synthesise a winning strategy for some bound k, then we can conclude that this strategy will also guarantee success for all larger bounds and therefore also in the unbounded case. Conversely, having no winning strategy for some k does not allow us to conclude that such a strategy does not exist in the unbounded case. General completeness of bounded model checking can be established by determining the completeness threshold of the problem instance and by setting k to this threshold. For the reachability properties that we consider, completeness thresholds are linear in the size of the state space [30].

Algorithm 1: Coalition-Versus-Opposition-Strategy-Synthesis($M, \langle\langle A\rangle\rangle \varphi, k$).

```
B := Agt\A,  Σ := {β^greedy}
loop forever do
    if sat([M, ⟨⟨A, Σ⟩⟩ φ, k]) for some assignment α then
        skip /*α succeeds against all strategies in Σ*/
    else
        return '[M, s0 ⊭_k ⟨⟨A⟩⟩ φ'
    if sat([M, ⟨⟨B, {α}⟩⟩ ¬φ, k]) for some assignment β then
        Σ := Σ ∪ {β} /*β succeeds against α*/
    else
        return '[M, s0 ⊨_k ⟨⟨A⟩⟩ φ] and α is a universally winning strategy'
```

6. Optimal and Nash equilibrium strategy synthesis

For a given MRA $M = (Agt, Res, d, Acc)$, a special case of the coalition versus opposition strategic bounded model checking problem is the *grand coalition* bounded model checking problem, which is to determine whether the grand coalition Agt has a joint winning strategy $\alpha_{Agt} = (\alpha_a)_{a \in Agt}$ to achieve the goal $\varphi = \bigwedge_{a \in Agt} (\mathbf{F} a.goal)$ within k steps. Formally, this problem is defined as follows:

$$[M, s_0 \models_k \langle\langle Agt\rangle\rangle \varphi] \;=\; \exists \alpha_{Agt} : [M, \pi(s_0, \alpha_{Agt}) \models_k \varphi]$$

where $\pi(s_0, \alpha_{Agt})$ is the execution path starting in the initial state s_0 where the grand coalition follows the strategy α_{Agt}. Since there is no opposition in the grand coalition model checking problem, following a joint strategy α_{Agt} will always result in exactly one execution path.

With our approach introduced in the previous sections we are able to solve the above problem and synthesise a winning strategy if existent. Such a strategy will ensure that the goal φ will be achieved. However, in several fields of application it may be of interest to find a strategy that allows to achieve the goal in an *optimal* way with regard to some criterion. A strategy of a single agent may be considered as optimal if it enables the agent to achieve its individual goal as early as possible, or as often as possible within k steps. Such optimisation criteria can be straightforwardly transferred to the grand coalition and its collective goal.

6.1. Optimal strategy synthesis

Subsequently, we extend our approach to the synthesis of *optimal winning strategies*. For this, we first define functions for quantifying the 'pay-off' of states and paths with regard to certain optimisation criteria.

Definition 25 *(State pay-off).* Let M be an MRA and let $a \in Agt$ be an agent. Then the *pay-off* for agent a in states $s \in S$ of M is a function $\rho_a : S \to \{0, 1\}$ where

$$\rho_a(s) \;=\; \begin{cases} 1 & \text{if } |s^{-1}(a)| = d(a), \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the pay-off of an agent is 1 in exactly the states where the demand of the agent is satisfied, i.e., the individual resource allocation goal is achieved. Assuming that we want to maximise the frequency of reaching the goal, the pay-off function can be extended to paths as follows:

Definition 26 *(Path pay-off – frequency).* Let M be an MRA, let $a \in Agt$ be an agent, and let $k \in \mathbb{N}$ be a bound. Then the k-bounded *frequency pay-off* for agent a on paths $\pi \in \Pi$ of M is a function $\rho_a^{fr} : \Pi \to \mathbb{N}$ where

$$\rho_a^{fr}(\pi) \;=\; \sum_{t=0}^{k} \rho_a\big(\pi[t]\big)$$

and $\pi[t]$ denotes the t-st state along π. The collective frequency pay-off of the grand coalition Agt is

$$\rho_{Agt}^{fr}(\pi) = \sum_{a \in Agt} \rho_a^{fr}(\pi).$$

Thus, a winning strategy can be regarded as collectively optimal with regard to frequency if following the strategy results in an execution path for which the collective frequency pay-off is maximal. An alternative criterion for optimality is speed, i.e., we wish to minimise the number of steps until the agents reach their goal for the first time. Before we can define speed pay-off, we first need an auxiliary function that yields the first position along a path where an agent reaches its goal.

Definition 27 *(First goal-achievement position).* Let M be an MRA, let $a \in Agt$ be an agent, and let $k \in \mathbb{N}$ be a bound. Then the k-bounded *first goal-achievement position* of agent a along a path $\pi \in \Pi$ in M is defined by the function $fgp_a : \Pi \to \mathbb{N}$ where

$$fgp_a(\pi) \;=\; \mathbf{min}\big(t \in [0, k+1] \,||\pi[t]^{-1}(a)| = d(a) \;\vee\; t = k+1\big)$$

and $\pi[t]$ denotes the t-st state along π.

Hence, $fgp_a(\pi)$ returns the earliest position along the k-prefix of π where agent a achieves its resource goal, or $fgp_a(\pi)$ returns $k+1$ if the agent does not achieve its goal along the k-prefix of π. Now the speed pay-off of a path can be defined as follows:

Definition 28 *(Path pay-off – speed).* Let M be an MRA, let $a \in Agt$ be an agent, and let $k \in \mathbb{N}$ be a bound. Then the k-bounded *speed pay-off* for agent a on paths $\pi \in \Pi$ of M is a function $\rho_a^{sp} : \Pi \to \mathbb{N}$ where

$$\rho_a^{sp}(\pi) \;=\; k - \big(fgp_a(\pi) - 1\big),$$

and the collective speed pay-off of the grand coalition Agt is

$$\rho_{Agt}^{sp}(\pi) = \sum_{a \in Agt} \rho_a^{sp}(\pi).$$

The earlier an agent a achieves its goal along the k-prefix of π, the higher its speed pay-off is. And, if the agent does not achieve its goal along the k-prefix, then the speed pay-off is 0.

We can now formally define collectively optimal strategies with regard to frequency or speed:

Definition 29 *(Collectively optimal strategy).* Let M be an MRA, let $k \in \mathbb{N}$ be a bound, and let φ be a goal-reachability formula. Moreover, let α_{Agt} be a joint strategy of the grand coalition and let $(\rho_a^c)_{a \in Agt}$ with $c \in \{fr, sp\}$ be a tuple of path pay-off functions, one for each $a \in Agt$. Then α_{Agt} is a *collectively optimal winning strategy* with regard to $(\rho_a^c)_{a \in Agt}$ if the following conditions hold:

1. $[M, \pi(s_0, \alpha_{Agt}) \models_k \varphi]$
2. $\neg\exists \alpha'_{Agt}$ with $[M, \pi(s_0, \alpha'_{Agt}) \models_k \varphi]$ and $\rho^c_{Agt}(\pi(s_0, \alpha'_{Agt})) > \rho^c_{Agt}(\pi(s_0, \alpha_{Agt}))$

where $\rho^c_{Agt}(\pi) = \sum_{a \in Agt} \rho^c_a(\pi)$.

Hence, a winning strategy α_{Agt} is collectively optimal with regard to a given tuple of pay-off functions if there does not exist an alternative winning strategy α'_{Agt} that results in a greater collective pay-off. Given a grand coalition bounded model checking problem $[M, s_0 \models_k \langle\langle Agt\rangle\rangle \varphi]$ and an optimisation criterion $c \in \{fr, sp\}$, we denote the corresponding collectively optimal strategy synthesis problem by $[M, s_0 \models_k \langle\langle Agt\rangle\rangle \varphi]^c_{Opt}$.

A collectively optimal winning strategy ensures that each agent achieves its goal at least once (Condition 1) and it additionally maximises the *overall* pay-off (Condition 2), but it does not necessarily optimise the pay-off of each individual agent in the grand coalition. Thus, such a strategy may favour certain agents while disadvantaging others.

6.2. Nash equilibrium strategy synthesis

Given a tuple of pay-off functions, it may be practically more useful and fair to synthesise a strategy that is a *Nash equilibrium* [9] rather than just being collectively optimal. In a joint Nash equilibrium strategy $\alpha_{Agt} = (\alpha_a)_{a \in Agt}$, no agent can increase its individual pay-off by deviating from α_{Agt}, assuming that all other agents keep following their local strategies α_a prescribed by α_{Agt}. A sub problem of the Nash equilibrium strategy synthesis problem is what we call the *single-agent strategy synthesis problem*. Given an MRA M, a joint strategy $\alpha_{Agt} = (\alpha_a)_{a \in Agt}$, a particular agent $a \in Agt$, and a goal φ, this problem is defined as follows:

$$[M, s_0 \models_k \langle\langle a | \alpha_{Agt \setminus a}\rangle\rangle \varphi] \;=\; \exists \alpha'_a : [M, \pi(s_0, \alpha_{Agt}[\alpha_a \leftarrow \alpha'_a]) \models_k \varphi]$$

where $\langle\langle a | \alpha_{Agt \setminus a}\rangle\rangle$ is the short-hand notation for $\langle\langle \{a\}, \{\alpha_{Agt \setminus a}\}\rangle\rangle$, $\alpha_{Agt \setminus a} = \alpha_{Agt} \setminus (\alpha_a)$, and $\alpha_{Agt}[\alpha_a \leftarrow \alpha'_a]$ is the substitution of α_a by α'_a in α_{Agt}. The single-agent strategy synthesis is also a special case of the strategic bounded model checking problem introduced in earlier sections. It asks whether the agent a has a strategy α'_a to achieve the collective goal φ, assuming that all other agents follow their strategies prescribed by α_{Agt}. We make use of the (negated) single-agent strategy synthesis problem in the definition of Nash equilibrium strategies:

Definition 30 *(Nash equilibrium strategy).* Let M be an MRA, let $k \in \mathbb{N}$ be a bound, and let φ be a goal-reachability formula. Moreover, let α_{Agt} be a joint strategy of the grand coalition and let $(\rho_a^c)_{a \in Agt}$ with $c \in \{fr, sp\}$ be a tuple of path pay-off functions, one for each $a \in Agt$. Then α_{Agt} is a *Nash equilibrium winning strategy* with regard to $(\rho_a^c)_{a \in Agt}$ if the following conditions hold:

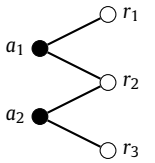
1. $[M, \pi(s_0, \alpha_{Agt}) \models_k \varphi]$
2. $\forall a \in Agt\ \neg\exists \alpha'_a$ with $[M, \pi(s_0, \alpha_{Agt}[\alpha_a \leftarrow \alpha'_a]) \models_k \varphi]$ and $\rho_a^c(\pi(s_0, \alpha_{Agt}[\alpha_a \leftarrow \alpha'_a])) > \rho_a^c(\pi(s_0, \alpha_{Agt}))$

Hence, a joint strategy α_{Agt} is a Nash equilibrium with regard to a tuple of pay-off functions $(\rho_a^c)_{a \in Agt}$ if it ensures that the collective goal will be achieved (Condition 1) and no agent can improve its individual pay-off by deviating from α_{Agt} with an alternative strategy α'_a, assuming that all other agents keep following their strategies prescribed by α_{Agt} (Condition 2). Given a grand coalition bounded model checking problem $[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi]$ and an optimisation criterion $c \in \{fr, sp\}$, we denote the corresponding Nash equilibrium strategy synthesis problem by $[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi]^c_{Nash}$.

6.3. Examples of winning, optimal, and Nash equilibrium strategies

We conclude this section with illustrative examples of winning, optimal and Nash equilibrium strategies of the grand coalition.

Example 3. The graph on the right describes an MRA M consisting of the agents a_1, a_2 and the resources r_1, r_2, r_3. The edges of the graph characterise the accessibility function. Moreover, we assume that the demand function of M is defined as follows: $d(a_1) = 2, d(a_2) = 2$.



We will exemplarily solve the grand coalition bounded model checking problem $[M, s_0 \models_6 \langle\langle Agt \rangle\rangle \varphi]$ for this MRA, i.e., we will check whether there exists a winning strategy that ensures that both agents will achieve their resource goals within 6 steps. Moreover, we will solve the optimal strategy synthesis and Nash equilibrium strategy synthesis generalisations of this problem. We can synthesise a winning strategy α^1_{Agt} that results in an execution path π_1 on which both agents will eventually reach their demand goal within at most 6 steps. The 6-prefix of π_1 is depicted in Fig. 1 (a). The states along the prefix are tuples of the form a_x, a_y, a_z where a_x denotes the agent that currently holds resource r_1, a_y denotes the agent that holds r_2, and a_z denotes the agent that holds r_3. Whenever x, y or z are 0, this denotes that the corresponding resource is currently unallocated. The transitions of the prefix are labelled with the joint strategic decisions of the form $\langle act_{a_1}, act_{a_2} \rangle$ that cause the change of states. Hence, the transition labels characterise the crucial part of the joint strategy α^1_{Agt} that results in the path π_1. We can see that α^1_{Agt} is a winning strategy for our grand coalition bounded model checking problem because it ensures that agent a_1 achieves its goal at time step 2 and agent a_2 achieves its goal at step 4. The achievement of goals is indicated by boxes in the corresponding states.

If we consider the frequency-optimisation variant of the above model checking problem, then α^1_{Agt} is not a collectively optimal strategy. It only allows each agent to reach its goal once within 6 steps, but it is in fact possible to make strategic decisions such that at least one of the agents will achieve its goal twice within 6 steps. A corresponding strategy α^2_{Agt} and 6-prefix of an execution path π_2 are shown in Fig. 1 (b). As it can be seen, agent a_1 reaches its goal at the steps 2 and 6, and agent a_2 reaches its goal at step 4. Hence, the frequency pay-off of the 6-prefix of this path is $\rho^{fr}_{Agt}(\pi_2) = \rho^{fr}_{a_1}(\pi_2) + \rho^{fr}_{a_2}(\pi_2) = 2 + 1 = 3$. The joint strategy α^2_{Agt} is a collectively optimal winning strategy with regard to frequency because there exists no alternative winning strategy that results in a pay-off higher than 3 for a bound of 6.

The strategy α^2_{Agt} favours agent a_1 over a_2. Hence, in order to determine whether α^2_{Agt} is a Nash equilibrium, we will check whether a_2 can deviate from α^2_{Agt} and improve its individual pay-off, assuming that a_1 still makes the strategic decisions prescribed by α^2_{Agt}. A possible deviation by a_2 is to request resource r_2 instead of resource r_3 in the initial state. This results in the alternative overall strategy α^3_{Agt} and the execution path prefix π_3 shown in Fig. 1 (c). As it can be seen, this strategy improves agent a_2's individual pay-off at the cost of a_1's pay-off, but the collective pay-off is still the same. However, in contrast to α^2_{Agt} the joint strategy α^3_{Agt} is a Nash equilibrium. There is no way for agent a_1 to unilaterally deviate from this strategy and increase its pay-off. All possible changes of strategic decisions by a_1 will result in either the same or even a lower pay-off. In the above example we have that the strategy α^3_{Agt} is a Nash equilibrium and at the same time collectively optimal. In general, the set of collectively optimal strategies and the set of Nash equilibrium strategies of a model checking problem may intersect but may also have distinct elements, whereas the set of winning strategies is always a superset of both the collectively optimal and the Nash equilibrium strategies.

In the subsequent section we will show that both the synthesis of collectively optimal and of Nash equilibrium strategies can be reduced to solving the maximum satisfiability problem of Boolean formulas.

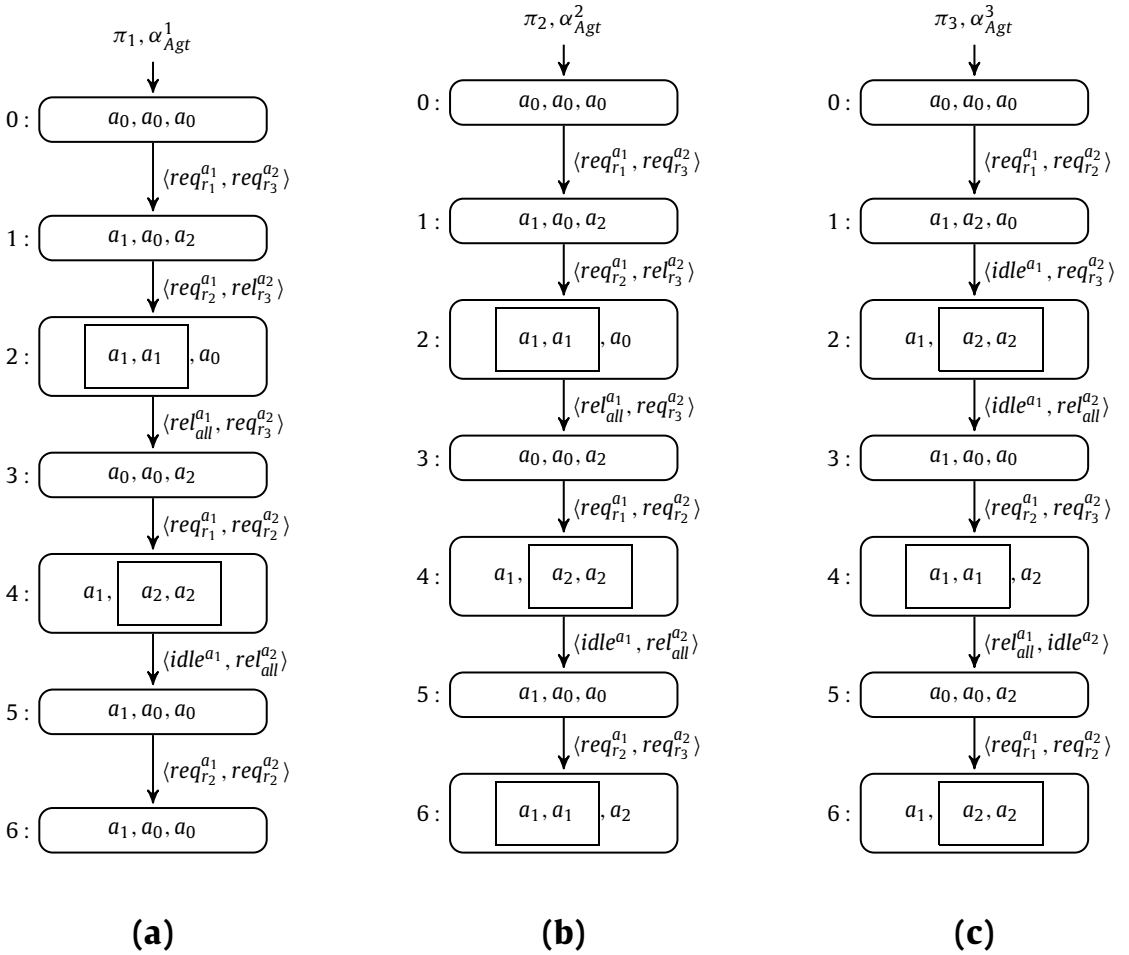


Fig. 1. (a): 6-prefix of a path π_1 corresponding to the winning but non-optimal strategy α^1_{Agt}. **(b):** Path π_2 corresponding to the winning strategy α^2_{Agt} that is collectively optimal with regard to frequency. **(c):** Path π_3 corresponding to the Nash equilibrium strategy α^3_{Agt}.

7. Max-SAT encoding of optimal and Nash equilibrium strategy synthesis

In this section we will show how the problems of synthesising collectively optimal and Nash equilibrium strategies can be encoded in propositional logic and solved via maximum-satisfiability (Max-SAT) solving. Both optimal and Nash equilibrium strategy synthesis are generalisations of the grand coalition bounded model checking problem

$$[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi].$$

We can straightforwardly encode this (non-generalised) problem in propositional logic based on the definitions from Section 4. We obtain the formula

$$\mathcal{F}_{Agt} \;=\; [\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k]$$

where $[\langle\langle Agt \rangle\rangle, k]$ encodes that all agents must follow a uniform strategy and adhere to the protocol (Definition 17), $[M, k]$ encodes all k-bounded paths of M (Definition 13) and $[\varphi, k]$ restricts the paths to those that satisfy φ (Definition 16).

From Theorem 1 in Section 4.3 we get that $\mathbf{sat}(\mathcal{F}_{Agt}) = \mathbf{1}$ if and only if the encoded grand coalition model checking problem holds. Moreover, from Theorem 2 we get that each truth assignment α that satisfies $\mathcal{F}_{Agt}$ characterises a winning strategy α_{Agt} of the grand coalition. The theorem also describes how α_{Agt} can be derived from α. Since we have a one-to-one correspondence between truth assignments and strategies, each assignment can be regarded as a strategy and vice versa.

7.1. Max-SAT-based optimal strategy synthesis

In order to synthesise an *optimal* winning strategy for the grand coalition we will extend the formula $\mathcal{F}_{Agt}$ with additional clauses that have weights $w \in \mathbb{N}_\infty$. We will use the weights to represent the pay-offs of agents in states of the system. By adding weighted clauses we will obtain a formula in weighted conjunctive normal form:

Definition 31 *(Weighted conjunctive normal form (WCNF)).* Let Var be a set of Boolean variables. A propositional logic formula $\mathcal{F}$ over Var in weighted conjunctive normal form is a conjunction of weighted clauses $(\mathcal{C}, w_\mathcal{C})$ where $\mathcal{C}$ is a standard clause and $w_\mathcal{C} \in \mathbb{N}_\infty$ is its weight. A clause $(\mathcal{C}, w_\mathcal{C})$ with $w_\mathcal{C} \in \mathbb{N}$ is called a soft clause and a clause $(\mathcal{C}, \infty)$ is called a hard clause.

For the sake of simplicity we typically just write $\mathcal{C}$ for hard clauses $(\mathcal{C}, \infty)$. Each WCNF formula $\mathcal{F}$ can be written as a conjunction $\mathcal{H} \wedge \mathcal{S}$ where $\mathcal{H}$ are the hard clauses and $\mathcal{S}$ are the soft clauses of $\mathcal{F}$.

For WCNF formulas the following optimisation problem has been defined:

Definition 32 *(Maximum satisfiability problem).* Let $\mathcal{F} = \mathcal{H} \wedge \mathcal{S}$ over Var be a propositional logic formula in weighted conjunctive normal form where $\mathcal{H}$ are the hard clauses and $\mathcal{S}$ are the soft clauses. The maximum satisfiability problem with regard to $\mathcal{F}$ is the problem of finding a truth assignment $\alpha : Var \to \{\mathbf{0}, \mathbf{1}\}$ that maximises

$$\sum_{(\mathcal{C}, w_\mathcal{C}) \in \mathcal{S}} \alpha(\mathcal{C}) \cdot w_\mathcal{C}$$

subject to the condition that $\alpha(\mathcal{H}) = \mathbf{1}$ holds.

Hence, the solution of the maximum satisfiability problem with regard to $\mathcal{F}$ is a truth assignment α that *maximises* the sum of weights of the satisfied soft clauses, under the condition that *all* hard clauses are satisfied. If no such assignment exists, then the maximum satisfiability problem has no solution.

For WCNF formulas $\mathcal{F} = \mathcal{H} \wedge \mathcal{S}$ over a set of variables Var and assuming that $\mathcal{A}(Var)$ is the set of all possible truth assignments over Var we define maximum satisfiability as the function

$$\textbf{max-sat}(\mathcal{F}) = \begin{cases} nil & \text{if } \textbf{sat}(\mathcal{H}) = \mathbf{0}, \\ \underset{\alpha \in \mathcal{A}(Var)}{\textbf{arg max}} \Big(\alpha(\mathcal{H}) \cdot \big(\sum\limits_{(\mathcal{C}, w_\mathcal{C}) \in \mathcal{S}} \alpha(\mathcal{C}) \cdot w_\mathcal{C} \big) \Big) & \text{otherwise.} \end{cases}$$

Hence, **max-sat**$(\mathcal{F})$ returns *nil* if the problem has no solution. Otherwise it returns the truth assignment satisfying all hard clauses that maximises the sum of weights of the satisfied soft clauses.

In our Max-SAT-based approach there will be a one-to-one correspondence between the path pay-off resulting from following a strategy α_{Agt} and the sum of weights of soft clauses satisfied by an assignment α. Therefore, we also define the pay-off as a function ρ on truth assignments and WCNF formulas:

$$\rho(\alpha, \mathcal{F}) = \alpha(\mathcal{H}) \cdot \big(\sum_{(\mathcal{C}, w_\mathcal{C}) \in \mathcal{S}} \alpha(\mathcal{C}) \cdot w_\mathcal{C} \big).$$

As it can be seen, the pay-off of assignments α that do not satisfy all hard clauses $\mathcal{H}$ will be always 0, and the pay-off of assignments that satisfy all hard clauses will be the sum of weights of satisfied soft clauses.

The extension of the formula $\mathcal{F}_{Agt}$ to a weighted formula $\mathcal{F}^c_{Agt}$ that encodes optimal strategy synthesis with regard to the criterion $c \in \{fr, sp\}$ consists of two additional components. The first component $[Aux^c_{Agt}, k]$ introduces auxiliary variables that allow to encode each goal-achievement sub formula of the form $[a.goal]_t$ (see Definition 21) as a unit clause consisting of a single Boolean variable g^a_t. The second component $[Opt^c_{Agt}, k]$ makes use of the auxiliary variables and encodes the actual optimisation problem with regard to the criterion c. Since the extensions for frequency optimisation and for speed optimisation differ, we discuss them separately in the subsequent sub sections.

7.1.1. Encoding frequency optimisation

For optimising the *frequency* pay-off, the auxiliary component to be added to the encoding is defined as follows:

Definition 33 *(Auxiliary encoding – frequency).* Let M be an MRA and let $k \in \mathbb{N}$. Then the k-bounded auxiliary encoding for frequency optimisation is

$$[Aux^{fr}_{Agt}, k] = \bigwedge_{a \in Agt} \bigwedge_{t=0}^{k} \big(g^a_t \leftrightarrow [a.goal]_t \big)$$

where g_t^a with $a \in Agt$ and $0 \leq t \leq k$ are the auxiliary variables introduced for the encoding, and $[a.goal]_t$ is defined according to the encoding of goals (Section 4.2).

If we conjunctively add the frequency auxiliary encoding to the overall encoding $\mathcal{F}_{Agt}$ of a grand coalition bounded model checking problem, then a truth assignment α will set an auxiliary variable g_t^a to *true* if and only if α corresponds to a grand coalition strategy α_{Agt} and $\pi(s_0, \alpha_{Agt})$ is an execution path where agent a reaches its goal at time step t. Hence, g_t^a is a *single-variable encoding* of a reaching its goal at step t. We can now use each g_t^a as a *unit clause* in an extension of our encoding and also add a weight to it, which we do in the frequency optimisation encoding:

Definition 34 *(Frequency optimisation encoding).* Let M be an MRA and let $k \in \mathbb{N}$. Then the k-bounded frequency optimisation encoding for an individual agent $a \in Agt$ is

$$[Opt_a^{fr}, k] = \bigwedge_{t=0}^{k} (g_t^a, \mathbf{1})$$

and the k-bounded frequency optimisation encoding for the grand coalition Agt is

$$[Opt_{Agt}^{fr}, k] = \bigwedge_{a \in Agt} [Opt_a^{fr}, k]$$

where g_t^a with $a \in Agt$ and $0 \leq t \leq k$ are the Boolean variables introduced in the frequency auxiliary encoding.

Hence, $[Opt_{Agt}^{fr}, k]$ consists of soft clauses, each with a weight of 1 and each clause encodes that an agent achieved its individual goal at a certain time step. The overall encoding of the problem of synthesising a collectively optimal strategy with regard to frequency is $\mathcal{F}_{Agt}^{fr} = \mathcal{F}_{Agt} \wedge [Aux_{Agt}^{fr}, k] \wedge [Opt_{Agt}^{fr}, k]$ where the sub formula $\mathcal{F}_{Agt} \wedge [Aux_{Agt}^{fr}, k]$ consists of hard clauses only. Solving **max-sat**$(\mathcal{F}_{Agt}^{fr})$ will return a truth assignment that satisfies all hard clauses and maximises the number of satisfied soft clauses, if such an assignment exists. In Section 7.1.3 we will show that such an assignment corresponds to a winning strategy that maximises the collective frequency pay-off.

7.1.2. Encoding speed optimisation

For optimising the *speed* pay-off, the auxiliary component to be added to the encoding is defined as follows:

Definition 35 *(Speed auxiliary encoding).* Let M be an MRA and let $k \in \mathbb{N}$. Then the k-bounded speed auxiliary encoding is

$$[Aux^{sp}, k] = \bigwedge_{a \in Agt} \bigwedge_{t=0}^{k} \left(fg_t^a \leftrightarrow ([a.goal]_t \wedge \bigwedge_{j=0}^{t-1} \neg [a.goal]_j)\right)$$

where fg_t^a with $a \in Agt$ and $0 \leq t \leq k$ are the auxiliary variables introduced for the encoding, and $[a.goal]_t$ is defined according to the encoding of goals (Section 4.2).

If we conjunctively add the speed auxiliary encoding to the overall encoding of a grand coalition bounded model checking problem, then a truth assignment α will set a variable fg_t^a to *true* if and only if α corresponds to a grand coalition strategy α_{Agt} and $\pi(s_0, \alpha_{Agt})$ is an execution path where agent a reaches its goal at time step t *for the first time* along $\pi(s_0, \alpha_{Agt})$. Hence, fg_t^a is a *single-variable encoding* of a reaching the goal at step t for the first time. We can now use each fg_t^a as a *unit clause* in an extension of our encoding and also add a weight to it, which we do in the speed optimisation encoding.

Definition 36 *(Speed optimisation encoding).* Let M be an MRA and let $k \in \mathbb{N}$. Then the k-bounded speed optimisation encoding for an individual agent $a \in Agt$ is

$$[Opt_a^{sp}, k] = \bigwedge_{t=0}^{k} \left(fg_t^a, \mathbf{(k\text{-}t)+1}\right)$$

and the k-bounded speed optimisation encoding for the grand coalition Agt is

$$[Opt_{Agt}^{sp}, k] = \bigwedge_{a \in Agt} [Opt_a^{sp}, k]$$

where fg_t^a with $a \in Agt$ and $0 \leq t \leq k$ are the Boolean variables introduced in the speed auxiliary encoding.

Hence, $[Opt^{sp}_{Agt}, k]$ consists of soft clauses where each clause encodes that an agent achieved its individual goal at a certain time step for the first time. The earlier the time step, the greater is the weight of the corresponding clause. The overall encoding of the problem of synthesising a collectively optimal strategy with regard to speed is $\mathcal{F}^{sp}_{Agt} = \mathcal{F}_{Agt} \wedge [Aux^{sp}_{Agt}, k] \wedge [Opt^{sp}_{Agt}, k]$ where the sub formula $\mathcal{F}_{Agt} \wedge [Aux^{sp}_{Agt}, k]$ consists of hard clauses only. Solving **max-sat**$(\mathcal{F}^{sp}_{Agt})$ will return a truth assignment that satisfies all hard clauses and maximises the sum of weights of the satisfied soft clauses, if such an assignment exists. In the next sub section we will show that such an assignment corresponds to a winning strategy that maximises the collective speed pay-off.

7.1.3. Properties of the optimal strategy synthesis encoding

Given a collectively optimal strategy synthesis problem $[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi]^c_{Opt}$ where $c \in \{fr, sp\}$, we can construct the corresponding WCNF encoding $\mathcal{F}^c_{Agt} = \mathcal{H}^c_{Agt} \wedge \mathcal{S}^c_{Agt}$ where $\mathcal{H}^c_{Agt} = [\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k] \wedge [Aux^c_{Agt}, k]$ are the hard clauses and $\mathcal{S}^c_{Agt} = [Opt^c_{Agt}, k]$ are the soft clauses of the encoding. For this encoding the following theorem holds:

Theorem 3 *(Optimal strategy synthesis). Let $[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi]^c_{Opt}$ be an optimal strategy synthesis problem and let $\mathcal{F}^c_{Agt}$ be its WCNF encoding. Then the following properties hold:*

1. *If **max-sat**$(\mathcal{F}^c_{Agt}) = nil$, then there does not exist a winning strategy for the grand coalition Agt to achieve φ within k steps.*
2. *If **max-sat**$(\mathcal{F}^c_{Agt}) = \alpha$, and α_{Agt} is the joint strategy corresponding to α, derived according to Theorem 2, then α_{Agt} is a collectively optimal winning strategy with regard to the criterion c.*

Proof. We have $\mathcal{F}^c_{Agt} = \mathcal{H}^c_{Agt} \wedge \mathcal{S}^c_{Agt}$ where $\mathcal{H}^c_{Agt} = [\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k] \wedge [Aux^c_{Agt}, k]$ are the hard clauses and $\mathcal{S}^c_{Agt} = [Opt^c_{Agt}, k]$ are the soft clauses of the encoding. Moreover, the WCNF formula $\mathcal{F}^c_{Agt}$ is defined over a set of Boolean variables Var, and $\mathcal{A}(Var)$ is the set of all possible truth assignments over Var.

Proof of Property 1: **max-sat**$(\mathcal{F}^c_{Agt}) = nil$ implies **sat**$([\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k] \wedge [Aux^c_{Agt}, k]) = \mathbf{0}$ (Definition of **max-sat** for WCNF formulas). The sub formula $[Aux^c_{Agt}, k]$ only introduces auxiliary variables and sets them equivalent to goal reachability properties (Definition 33 and 35). Hence, $[Aux^c_{Agt}, k]$ cannot be the cause of unsatisfiability and we can conclude that **sat**$([\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k]) = \mathbf{0}$. This implies that $[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi]$ does not hold (Theorem 1). We can now immediately conclude that there does not exist a winning strategy for the grand coalition Agt to achieve φ within k steps.

Proof of Property 2: **max-sat**$(\mathcal{F}^c_{Agt}) = \alpha$ implies

$$\alpha = \underset{\alpha \in \mathcal{A}(Var)}{\mathbf{arg\,max}} \Big(\alpha(\mathcal{H}^c_{Agt}) \cdot \big(\sum_{(\mathcal{C}, w_{\mathcal{C}}) \in \mathcal{S}^c_{Agt}} \alpha(\mathcal{C}) \cdot w_{\mathcal{C}} \big) \Big)$$

(Definition of **max-sat** for WCNF formulas). Hence, the assignment α satisfies all hard clauses of $\mathcal{F}^c_{Agt}$. In particular, $\alpha([\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k]) = \mathbf{1}$. We can conclude that $[M, s_0 \models_k \langle\langle Agt \rangle\rangle \varphi]$ holds (Theorem 1), and that α characterises a corresponding winning strategy α_{Agt} (Theorem 2).

Now we still need to prove that α_{Agt} is collectively optimal with regard to the criterion c. We show this for $c = fg$, i.e., for the case of frequency optimisation. According to the definition of **max-sat**, out of all truth assignments that satisfy all hard clauses, α is the assignment that maximises the sum of weights of the satisfied soft clauses. Let w_α be the sum of weights of the soft clauses satisfied by α. Then for all other truth assignments α' that satisfy all hard clauses we have that $w_\alpha \geq w_{\alpha'}$. The assignment α makes w_α clauses of the sub formula $[Opt^{fr}_{Agt}, k]$ *true* where each clause of $[Opt^{fr}_{Agt}, k]$ is of the form $(g^a_t, \mathbf{1})$ for distinct agents $a \in Agt$ and time steps $0 \leq t \leq k$ (Definition 34). For each of these pairs of agents and time steps, $\alpha(g^a_t) = \mathbf{1}$ implies $\alpha([a.goal]_t) = \mathbf{1}$ (Definition 33). The hard clauses of the encoding, $[\langle\langle Agt \rangle\rangle, k] \wedge [M, k] \wedge [\varphi, k]$, ensure that $\alpha([a.goal]_t) = \mathbf{1}$ if and only if α characterises a joint strategy α_{Agt} that results in a path $\pi(s_0, \alpha_{Agt})$ where agent a reaches its resource goal at time step t (Definitions 13, 14, 15). We can conclude that α characterises a strategy that results in a k-bounded path where w_α times some agent reaches its resource goal. Hence, the collective frequency pay-off of the path $\pi(s_0, \alpha_{Agt})$ is $\rho^{fr}_{Agt}(\pi(s_0, \alpha_{Agt})) = w_\alpha$ (Definition 26). Based on a similar argumentation, we can show that each alternative truth assignment α' that satisfies all hard clauses characterises a path $\pi(s_0, \alpha'_{Agt})$ with a pay-off $\rho^{fr}_{Agt}(\pi(s_0, \alpha'_{Agt})) = w_{\alpha'}$. We already argued that $w_\alpha \geq w_{\alpha'}$ for all alternative assignments α'. Thus, we can conclude that α characterises a joint winning strategy α_{Agt} that is collectively optimal with regard to frequency. The proof for the case of speed optimisation is analogous. □

7.2. Max-SAT-based Nash equilibrium strategy synthesis

We will now show how our approach to the synthesis of collectively optimal strategies can be extended to the synthesis of Nash equilibrium strategies. As discussed in Section 8, a sub problem of the Nash equilibrium strategy synthesis is the

single-agent strategy synthesis $[M, s_0 \models_k \langle\langle a|\alpha_{Agt\setminus a}\rangle\rangle \varphi]$. Given a joint strategy α_{Agt} and an agent a, the single-agent strategy synthesis is the problem of determining whether a has a strategy α'_a to achieve the collective goal φ, assuming that all other agents follow their strategies prescribed by α_{Agt}. This problem can be generalised to the synthesis of an *optimal* strategy for the single agent a with regard to a criterion c: $[M, s_0 \models_k \langle\langle a|\alpha_{Agt\setminus a}\rangle\rangle \varphi]^c_{Opt}$, i.e., the generalised problem is to find a strategy α'_a for agent a that maximises its individual pay-off, assuming that the remaining agents follow what α_{Agt} prescribes to them. Based on the definitions from the previous sub sections, we can encode this problem as a WCNF formula:

$$\mathcal{F}^c_{a|\alpha_{Agt\setminus a}} = [\langle\langle a\rangle\rangle, k] \wedge [\alpha_{Agt\setminus a}, k] \wedge [M, k] \wedge [\varphi, k] \wedge [Aux^c_a, k] \wedge [Opt^c_a, k].$$

Solving **max-sat**$(\mathcal{F}^c_{a|\alpha_{Agt\setminus a}})$ will yield a truth assignment from which we can derive an optimal strategy α'_a for agent a. Now, how does this help us to synthesise a joint strategy that is a Nash equilibrium? A joint winning strategy $\alpha_{Agt} = (\alpha_a)_{a\in Agt}$ is a Nash equilibrium if *no* agent $a \in Agt$ can deviate from α_{Agt} with an alternative strategy α'_a that would increase its pay-off, assuming that all other agents follow their strategies prescribed by α_{Agt}. Hence, if we can show for a joint strategy $\alpha_{Agt} = (\alpha_a)_{a\in Agt}$ that for each agent $a \in Agt$ the optimal strategy α'_a for the single-agent strategy synthesis problem $[M, s_0 \models_k \langle\langle a|\alpha_{Agt\setminus a}\rangle\rangle \varphi]^c_{Opt}$ is not better than α_a, then we can conclude that α_{Agt} is a Nash equilibrium. We developed a max-sat-based algorithm, Algorithm 2, that gradually modifies a joint strategy α_{Agt} until the strategy becomes a Nash equilibrium or a cycle of non-equilibrium strategies is detected.

Algorithm 2: Nash-Equilibrium-Strategy-Synthesis($\mathcal{F}^c_{Agt}$).

```
1  α^0_Agt := max-sat(F^c_Agt)
2  if (α^0_Agt = nil) then
3  |  return 'no winning strategy exists'
4  for i = 0 to ∞ do
5  |  for each a ∈ Agt do
6  |  |  α^{i+1}_a := max-sat(F^c_{a|α^i_{Agt\a}})
7  |  |  if (ρ(α^{i+1}_a, F^c_{a|α^i_{Agt\a}}) > ρ(α^i_a, F^c_{a|α^i_{Agt\a}})) then
8  |  |  |  α^{i+1}_Agt := α^i_Agt[α^i_a ← α^{i+1}_a]
9  |  |  |  if (∃0 ≤ j < i : α^j_Agt = α^{i+1}_Agt) then
10 |  |  |  |  return 'no equilibrium detectable'
11 |  |  |  break
12 |  return α^i_Agt
```

The algorithm takes the encoding $\mathcal{F}^c_{Agt}$ of an optimal strategy synthesis problem $[M, s_0 \models_k \langle\langle Agt\rangle\rangle \varphi]^c_{Opt}$ as an input. It first generates a collectively optimal winning strategy α^0_{Agt}, if existent (Line 1). If this is not the case, then no winning strategy exists, and therefore, also no Nash equilibrium strategy exists. Otherwise, α^0_{Agt} gets iteratively revised to a strategy α^{i+1}_{Agt} where i indicates the iteration. In each iteration i it is checked for each agent $a \in Agt$ whether its current individual strategy α^i_a prescribed by the joint strategy α^i_{Agt} is optimal or whether an alternative individual strategy α^{i+1}_a exists that grants a a greater pay-off. This step involves to solve the single-agent optimal strategy synthesis problem for agent a (Line 6). In case such a better strategy for agent a exists (Line 7), then α^i_a is replaced by α^{i+1}_a in the joint strategy (Line 8) and the algorithm enters the next iteration (Line 11). If an iteration i is reached where *none* of the agents can improve its pay-off by following an alternative strategy, then the current joint strategy α^i_{Agt} is a Nash equilibrium. The algorithm then terminates and returns α^i_{Agt} (Line 12). A further condition for termination is that a cycle of non-equilibrium strategies has been generated, which implies that no Nash equilibrium is detectable (Lines 9,10). In each iteration of the algorithm a strategy is generated that either differs from all previously considered strategies or is identical to some previously generated strategy. The detection of a strategy that is identical to a previously generated strategy results in immediate termination. Since for finite-state MRAs only finitely many different strategies exist, the eventual termination of the algorithm is guaranteed – although for larger MRAs an exhaustive exploration of the strategy space may be unrealistic.

Theorem 4 *(Nash equilibrium strategy synthesis). Let* $[M, s_0 \models_k \langle\langle Agt\rangle\rangle \varphi]^c_{Nash}$ *be a Nash equilibrium strategy synthesis problem and let* $\mathcal{F}^c_{Agt}$ *be the WCNF encoding of the corresponding optimal strategy synthesis problem. If Nash-Equilibrium-Strategy-Synthesis*$(\mathcal{F}^c_{Agt})$ *returns a joint strategy* α^i_{Agt}*, then* α^i_{Agt} *is a Nash equilibrium with regard to the criterion* c*.*

Proof. The algorithm will only enter the iterative loop if a winning strategy exists. According to the definition of the single-agent strategy synthesis problem, each modification of the strategy based on the lines 6 and 8 will still result in a winning strategy. The lines 5-7 ensure that the algorithm will only return the current strategy α^i_{Agt} if *no* agent a can

```
agents:
  - a1
  - a2

resources:
  - r1
  - r2
  - r3

coalition:
  - a1

bound: 10

a1:
  demand: 2
  access:
    - r1
    - r2

a2:
  demand: 2
  access:
    - r2
    - r3
```

Fig. 2. SATMAS input corresponding to Example 3.

deviate from α^i_{Agt} with an alternative individual strategy α^{i+1}_a that increases a's pay-off, assuming that the remaining agents still follow their individual strategy prescribed by α^i_{Agt}. Thus, according to Definition 30 the joint strategy α^i_{Agt} is a Nash equilibrium. □

Hence, the algorithm allows us to synthesise Nash equilibrium strategies for grand coalition bounded model checking problems if its execution does not run into a cycle of non-equilibrium strategies. We report on successful syntheses in the subsequent section. As future work, we plan to derive propositional logic constraints from detected non-equilibrium cycles and restart the algorithm with these constraints in order to increase the exhaustiveness of the search for an equilibrium.

8. Implementation and experiments

In this section we introduce the tool SATMAS[1] that implements our synthesis technique and we present experimental results. All experiments were conducted on a 2.6 GHz Intel Core i7 system with 16 GB.

8.1. SATMAS

We developed the tool SATMAS that implements our approach in Python. SATMAS takes a specification of a multi-agent system for resource allocation $M = (Agt, Res, d, Acc)$, a coalition $A \subseteq Agt$ and a bound k as an input. The text file-based input has an intuitive format which is exemplified in Fig. 2. Tool users can either manually specify input files or make use of a random MRA generator.

The strategic property to be checked is $\varphi = \langle\langle A \rangle\rangle \left(\bigwedge_{a \in A} \left(\mathbf{F} a.goal \right) \right)$, i.e., whether there exists a strategy that ensures that each agent in the coalition A achieves its goal (i.e., satisfies its demand) within k steps. The tool offers the option to check this property for the input k only, or to check the property iteratively for all bounds from 0 to k. Our tool supports the two modes *coalition versus opposition strategy synthesis* and *grand coalition optimal strategy synthesis*.

8.2. Experiments on coalition versus opposition strategy synthesis

The *coalition versus opposition strategy synthesis* builds the propositional logic encodings of the strategic bounded model checking problem and of its complement. Then it executes Algorithm 1 *Coalition-Versus-Opposition-Strategy-Synthesis*. The encoding process includes optimisations such as logical simplifications and the Tseitin transformation into conjunctive normal form. The state-of-the art solver CaDiCaL [10] is employed to check the satisfiability of the encodings and to determine satisfying truth assignments from which winning strategies can be derived. The solver is executed as a sub-process of our tool SATMAS. It receives the encoding in a DIMACS file and eventually sends the result of the satisfiability check back to

[1] available at github.com/TuksModelChecking/SATMAS.

Table 1
Coalition versus opposition strategy synthesis.

Scenario	Bound	Time
$\lvert A\rvert = 3$, $\lvert B\rvert = 3$, $\lvert Res\rvert = 6$, $\mathcal{D} = [2, 3]$, $\mathcal{ACC} = [4, 5]$	64	4.0*s*
$\lvert A\rvert = 3$, $\lvert B\rvert = 3$, $\lvert Res\rvert = 6$, $\mathcal{D} = [\mathbf{3}, \mathbf{4}]$, $\mathcal{ACC} = [4, 5]$	64	4.8*s*
$\lvert A\rvert = 3$, $\lvert B\rvert = 3$, $\lvert Res\rvert = 6$, $\mathcal{D} = [3, 4]$, $\mathcal{ACC} = [\mathbf{5}, \mathbf{6}]$	64	11.7*s*
$\lvert \boldsymbol{A}\rvert = \mathbf{4}$, $\lvert B\rvert = 3$, $\lvert \boldsymbol{Res}\rvert = \mathbf{8}$, $\mathcal{D} = [3, 4]$, $\mathcal{ACC} = [5, 6]$	64	15.1*s*
$\lvert A\rvert = 4$, $\lvert \boldsymbol{B}\rvert = \mathbf{4}$, $\lvert Res\rvert = 8$, $\mathcal{D} = [3, 4]$, $\mathcal{ACC} = [5, 6]$	64	42.5*s*
$\lvert A\rvert = 4$, $\lvert B\rvert = 4$, $\lvert Res\rvert = 8$, $\mathcal{D} = [\mathbf{5}, \mathbf{6}]$, $\mathcal{ACC} = [5, 6]$	64	45.0*s*
$\lvert A\rvert = 4$, $\lvert B\rvert = 4$, $\lvert Res\rvert = 8$, $\mathcal{D} = [5, 6]$, $\mathcal{ACC} = [\mathbf{6}, \mathbf{7}]$	64	96.7*s*
$\lvert \boldsymbol{A}\rvert = \mathbf{5}$, $\lvert B\rvert = 4$, $\lvert \boldsymbol{Res}\rvert = \mathbf{10}$, $\mathcal{D} = [5, 6]$, $\mathcal{ACC} = [6, 7]$	64	133*s*
$\lvert A\rvert = 5$, $\lvert \boldsymbol{B}\rvert = \mathbf{5}$, $\lvert Res\rvert = 10$, $\mathcal{D} = [5, 6]$, $\mathcal{ACC} = [6, 7]$	64	392*s*
$\lvert A\rvert = 5$, $\lvert B\rvert = 5$, $\lvert Res\rvert = 10$, $\mathcal{D} = [\mathbf{6}, \mathbf{8}]$, $\mathcal{ACC} = [\mathbf{8}, \mathbf{9}]$	64	958*s*

Table 2
Optimal strategy synthesis – increased accessibility.

Scenario	Bound	Pay-Off	Time
$\lvert Agt\rvert = 5$, $\lvert Res\rvert = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	25	3.7*s*
$\lvert Agt\rvert = 5$, $\lvert Res\rvert = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 3$	50	34	15.0*s*
$\lvert Agt\rvert = 5$, $\lvert Res\rvert = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 4$	50	34	146*s*
$\lvert Agt\rvert = 5$, $\lvert Res\rvert = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 5$	50	34	610*s*

SATMAS. In coalition versus opposition strategy synthesis experiments we were able to verify goal-reachability properties of MRAs with up to ten agents and ten resources within our pre-set time limit of 1000 seconds. A selection of the results where a winning strategy for the coalition A against the opposition B could be synthesised is shown in Table 1. The *Scenario* column indicates the size of A, the size of B and the overall number of resources Res. Moreover, $\mathcal{D}$ is the interval from which the demand of each agent was randomly selected, and $\mathcal{ACC}$ is the interval from which number of accessible resources of each agent was randomly selected. The column *Bound* indicates the fixed bound of $k = 64$ that was used in all experiments on the coalition versus opposition strategy synthesis. This chosen bound was sufficiently large to synthesize a winning strategy for each of the considered scenarios. The column *Time* shows the overall time that the tool spent on encoding and satisfiability solving. We increased certain parameters of the scenario in the different rows of the table in order to evaluate which parameters have the most significant impact on the time. The parameters that were increased in comparison to the previous scenario are highlighted in bold.

We detected that increasing the accessibility while keeping the remaining parameters unchanged involved the most significant increase of the synthesis time. A higher degree of accessibility means more possibilities for strategic decisions, which naturally leads to an enlarged search space. An increase of the size of the opposition has a similarly high impact on the synthesis time. More agents in the opposition mean that a winning strategy to be synthesised has to succeed against an enlarged set of strategies that the opposition may follow. The increase of the parameters *size of the coalition* and *amount of resources* has a recognisable effect on the synthesis time. Both parameters contribute to the size of the state space to be analysed, which is in $\mathcal{O}(|A + B|^{|Res|})$. Finally, we detected that the demand is the least significant contributor to the synthesis time. An increased demand neither increases the size of the state space nor the amount of possible strategic decisions.

8.3. Experiments on grand coalition optimal strategy synthesis

The *grand coalition optimal strategy synthesis* mode requires that the chosen coalition is the grand coalition Agt. Additionally, an optimisation criterion c, either frequency or speed, must be selected via command line options. In this mode our tool builds the logic encoding of the optimal strategy synthesis problem $[M, s_0 \models_k \varphi]^c_{Opt}$. The maximum-satisfiability solver OPEN-WBO [8] is employed to determine a truth assignment for the encoding that satisfies all hard clauses and maximises the sum of weights of the satisfied soft clauses. OPEN-WBO is one of the top-ranked solvers of the MaxSAT Evaluation 2022 [31]. From an optimal truth assignment the corresponding optimal strategy and the collective pay-off can be derived. In experiments on the grand coalition optimal strategy synthesis we used the well-known dining philosophers problem as a benchmark, which can be regarded as a special case of an MRA: Five philosophers (i.e., agents) sit at a round table where adjacent philosophers share a fork (i.e., resource), and each philosopher has a demand of two. The synthesis of a frequency-optimal strategy for this scenario takes 3.7 seconds with our tool. We generalised the original problem with regard to several parameters of the scenario and analysed the effect on the synthesis time.

Table 2 shows the experimental results for scenarios with an increasing accessibility of agents to resources. In the first scenario with $\mathcal{ACC} = 2$ agent a_1 has access to the resources r_1 and r_2, agent a_2 has access to r_2 and r_3, and so forth. In the second scenario with $\mathcal{ACC} = 3$ each agent has access to three adjacent resources. In the final scenario each agent has access to all five resources. The column *Pay-Off* indicates the collective pay-off that could be achieved within the bound.

Table 3
Optimal strategy synthesis – increased demand.

Scenario	Bound	Pay-Off	Time
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 5$	50	34	610*s*
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 3$, $\mathcal{ACC} = 5$	50	16	228*s*
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 4$, $\mathcal{ACC} = 5$	50	10	84.7*s*
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 5$, $\mathcal{ACC} = 5$	50	N/A	11.8*s*

Table 4
Optimal strategy synthesis – increased number of agents.

Scenario	Bound	Pay-Off	Time
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	25	3.7*s*
$\|Agt\| = 6$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	25	5.6*s*
$\|Agt\| = 7$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	25	9.1*s*
$\|Agt\| = 8$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	24	13.9*s*
$\|Agt\| = 9$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	22	105*s*
$\|Agt\| = 10$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	17	96.8*s*
$\|Agt\| = 11$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	N/A	16.2*s*

Table 5
Optimal strategy synthesis – increased number of resources.

Scenario	Bound	Pay-Off	Time
$\|Agt\| = 10$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	17	96.8*s*
$\|Agt\| = 10$, $\|Res\| = 6$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	25	25.7*s*
$\|Agt\| = 10$, $\|Res\| = 7$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	33	34.9*s*
$\|Agt\| = 10$, $\|Res\| = 8$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	48	47.7*s*
$\|Agt\| = 10$, $\|Res\| = 9$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	51	39.1*s*
$\|Agt\| = 10$, $\|Res\| = 10$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	66	38.4*s*

As it can be seen, the increasing accessibility at a constant demand level leads to a considerable growth of the synthesis time. Access to a higher number of resources involves significantly more possibilities to achieve the resource allocation goals which the solver has to consider.

Table 3 shows the experimental results for scenarios with an increasing demand of each agent. In each scenario each agent has access to all five resources.

Thus, the smaller the difference between the accessibility and the demand, the faster the synthesis problem can be solved. Clearly, if the accessibility and the demand are at a similar level then there are less resource allocation possibilities to be considered in the synthesis. In the latter row of Table 3 we have a scenario where no optimal *winning* strategy for a bound of 50 exists. As we can see, the non-existence of such a strategy can be shown quite fast.

In the next set of experiments we analysed the impact of the number of agents on the optimal strategy synthesis. We increased the number of agents while keeping the remaining scenario parameters constant. The access of the additional agents to resources was arranged such that each pair of agents shared at most one resource. The results are shown in Table 4.

In general, the synthesis time grows with the number of agents in the system. However, this growth is less significant than the growth induced by an increased accessibility (compare Table 2). The synthesis in the scenario with ten agents was even slightly faster than the synthesis in the scenario with nine agents. Our conjecture is that in certain cases adding further agents to a scenario can introduce significant constraints to the strategy synthesis problem to be solved. This may result in a constrained problem that is simpler and thus solvable faster. In the final scenario of Table 4 we have another case where no optimal *winning* strategy exists, which the solver quickly detected.

We extend the experiments with ten agents by scenarios with an increasing number of resources. In each scenario the access of agents to resources was evenly distributed. Thus, the final scenario with ten agents and ten resources represents a classical dining philosophers problem with a table capacity of ten. The results are shown in Table 5.

Although a higher number of resources involves a larger state space to be explored, this set of experiments did not reveal a clear correlation between the amount of resources in the system and the synthesis time. Our conjecture is that in certain scenarios additional resources make the synthesis problem even easier to be solved, since with additional resources less sharing is necessary. We attempted to extend the scenario with ten agents and ten resources by additionally increasing the accessibility. However, our tool was not able to solve this extended scenario within a 1000 seconds time frame. This shows again that the accessibility, or more specifically, the difference between the accessibility and the demand is the factor with the highest impact on the synthesis performance.

In a further set of experiments we evaluated the effect of the bound on the synthesis time. The results are shown in Table 6.

Table 6
Optimal strategy synthesis – increased bound.

Scenario	Bound	Pay-Off	Time
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	50	25	3.7*s*
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	100	50	9.4*s*
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	200	100	27.5*s*
$\|Agt\| = 5$, $\|Res\| = 5$, $\mathcal{D} = 2$, $\mathcal{ACC} = 2$	500	250	150*s*

Unsurprisingly, an increase of the bound also increases the synthesis time. It can be expected that if the agents follow an optimal strategy, the system will eventually end up in a loop of repetitive states – and the number of actually reached states may be significantly smaller than the bound. Hence, instead of working with large bounds it may be an alternative to extend the synthesis technique by a loop detection mechanism. This is left as future work.

8.4. Experiments on grand coalition Nash equilibrium strategy synthesis

We also prototypically implemented the *grand coalition Nash equilibrium strategy synthesis*. For several MRAs with up to four agents and four resources we were able to detect winning strategies that are Nash equilibria within one minute, whereas for other MRAs of the same size our tool timed out. A time out may indicate that no equilibrium exists. But such a conclusion can be only drawn if all possible strategies have been considered, which may be practically infeasible for many scenarios. Hence, so far our approach is capable of finding equilibria in small MRAs, but not of proving their non-existence. Our Nash equilibrium strategy synthesis leaves space for improvement in future work. Since the synthesis requires to solve a sequence of related Max-SAT problems, parallel solving with clause sharing may be applicable. Furthermore, heuristics could be employed to determine the order in which Algorithm 2 should iterate over the agents.

9. Conclusion and outlook

In this article we introduced a technique for synthesising winning, collectively optimal, and Nash equilibrium strategies for multi-agent systems for resource allocation (MRAs). The agents in these systems have once-off resource allocation goals that they pursue to achieve. Given a partition of the agents into a coalition and an opposition, a winning strategy is a plan of actions that ensures that all agents in the coalition achieve their goals, no matter how the opposition acts. Agents may have the incentive to achieve their goals in an optimal way, e.g., as frequent as possible or as early as possible. We defined pay-offs to measure optimality. An optimal strategy is a plan that guarantees that the overall pay-off of the collective of agents is maximal. A Nash equilibrium strategy is a plan that ensures that no agent can increase its individual pay-off by unilaterally deviating from the plan. Although MRAs are an abstract concept, several practically relevant resource allocation problems can be modelled by MRAs. Winning, optimal, and Nash equilibrium strategies can provide efficient and fair solutions to these problems.

Our technique is based on a reduction of the winning strategy synthesis problem to the Boolean satisfiability problem (SAT) and a reduction of the optimal strategy synthesis problem to the maximum satisfiability problem (Max-SAT). We showed that from a SAT resp. Max-SAT solution a winning resp. collectively optimal strategy can be immediately derived. Furthermore, we defined an algorithm that iteratively alters a collectively optimal strategy until it becomes a Nash equilibrium or until a cycle of non-equilibrium strategies is detected. We have implemented our technique on top of the solvers CaDiCaL and OPEN-WBO. In an experimental evaluation we analysed the capabilities and limitations of our approach.

As future work we plan to generalise our multi-agent systems for resource allocation, allowing different types of resources and allowing advanced goals such as allocating resources in a particular order or holding resources for a certain number of time steps. This will enable us to consider more realistic scenarios such as resource allocation in distributed operating systems [32], wireless sensor networks [33] or clouds [34]. We also intend to extend our approach to the synthesis of Pareto optimal strategies [35] which may be more useful than Nash equilibrium strategies in certain scenarios. Another direction of future work is to transfer our Max-SAT-based technique to the synthesis of optimal multi-agent systems [36]: Given a set of agents, a set of goals and a budget for purchasing access to resources, the most cost-efficient MRA for reaching all goals shall be synthesised. Further plans are the integration of partial order reduction [37] and symmetry reduction [38] in order to reduce the complexity of model checking and strategy synthesis. Moreover, we intend to combine our bounded model checking approach with k-induction [39] in order to enable unbounded model checking in an efficient way. In our present experiments, we experienced that if an encoded strategy synthesis problem could not be solved within 1000 seconds with the (Max-)SAT solvers that we used, then it was very unlikely that the problem could be solved in a significantly longer run of the solvers. Thus, we are also planning to re-run experiments on computer clusters and to integrate further solvers into our tool.

CRediT authorship contribution statement

Nils Timm: Conceptualization, Investigation, Supervision, Writing – review & editing. **Josua Botha:** Software. **Steven Jordaan:** Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. De Masellis, V. Goranko, S. Gruner, N. Timm, Generalising the dining philosophers problem: competitive dynamic resource allocation in multi-agent systems, in: European Conference on Multi-Agent Systems, Springer, 2018, pp. 30–47.
[2] W. Li, F.C. Delicato, P.F. Pires, Y.C. Lee, A.Y. Zomaya, C. Miceli, L. Pirmez, Efficient allocation of resources in multiple heterogeneous wireless sensor networks, J. Parallel Distrib. Comput. 74 (1) (2014) 1775–1788.
[3] R. Alur, T.A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (5) (2002) 672–713.
[4] R. Alur, T.A. Henzinger, F.Y. Mang, S. Qadeer, S.K. Rajamani, S. Tasiran, Mocha: Modularity in model checking, in: International Conference on Computer Aided Verification, Springer, 1998, pp. 521–525.
[5] A. Lomuscio, H. Qu, F. Raimondi, Mcmas: an open-source model checker for the verification of multi-agent systems, Int. J. Softw. Tools Technol. Transf. 19 (1) (2017) 9–30.
[6] N. Timm, J. Botha, Model checking and strategy synthesis for multi-agent systems for resource allocation, in: Brazilian Symposium on Formal Methods, Springer, 2021, pp. 53–69.
[7] P. Hansen, B. Jaumard, Algorithms for the maximum satisfiability problem, Computing 44 (4) (1990) 279–303.
[8] R. Martins, N. Manthey, M. Terra-Neves, V. Manquinho, I. Lynce, Open-wbo@ maxsat evaluation 2020, MaxSAT Eval. 2020 (2021) 24.
[9] J.F. Nash Jr, Equilibrium points in n-person games, Proc. Natl. Acad. Sci. 36 (1) (1950) 48–49.
[10] A. Biere, K. Fazekas, M. Fleury, M. Heisinger, CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020, in: T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, M. Suda (Eds.), Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions, in: Department of Computer Science Report Series B, vol. B-2020-1, University of Helsinki, 2020, pp. 51–53.
[11] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[12] A. Cimatti, E. Clarke, F. Giunchiglia, M. Roveri, Nusmv: a new symbolic model checker, Int. J. Softw. Tools Technol. Transf. 2 (4) (2000) 410–425.
[13] A. Biere, A. Cimatti, E.M. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Handb. Satisf. 185 (99) (2009) 457–481.
[14] F. Raimondi, A. Lomuscio, Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams, J. Appl. Log. 5 (2) (2007) 235–251.
[15] S. Schewe, Atl* satisfiability is 2exptime-complete, in: International Colloquium on Automata, Languages, and Programming, Springer, 2008, pp. 373–385.
[16] A. Lomuscio, W. Penczek, B. Woźna, Bounded model checking for knowledge and real time, Artif. Intell. 171 (16–17) (2007) 1011–1038.
[17] X. Luo, K. Su, A. Sattar, M. Reynolds, Verification of multi-agent systems via bounded model checking, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2006, pp. 69–78.
[18] M. Kacprzak, W. Penczek, Unbounded model checking for alternating-time temporal logic, in: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004, AAMAS 2004, IEEE, 2004, pp. 646–653.
[19] J. Pilecki, M.A. Bednarczyk, W. Jamroga, Smc: synthesis of uniform strategies and verification of strategic ability for multi-agent systems, J. Log. Comput. 27 (7) (2017) 1871–1895.
[20] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, A. Toumi, Rational verification: from model checking to equilibrium checking, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, 2016.
[21] J. Gutierrez, P. Harrenstein, M. Wooldridge, From model checking to equilibrium checking: reactive modules for rational verification, Artif. Intell. 248 (2017) 123–157.
[22] J. Gutierrez, M. Najib, G. Perelli, M. Wooldridge, On computational tractability for rational verification, in: International Joint Conferences on Artificial Intelligence, 2019.
[23] S. Almagor, O. Kupferman, G. Perelli, Synthesis of controllable Nash equilibria in quantitative objective game, in: IJCAI, vol. 18, 2018, pp. 35–41.
[24] J. Gutierrez, A. Murano, G. Perelli, S. Rubin, M. Wooldridge, Nash Equilibria in Concurrent Games with Lexicographic Preferences, 2017.
[25] J. Gutierrez, A. Murano, G. Perelli, S. Rubin, T. Steeples, M. Wooldridge, Equilibria for games with combined qualitative and quantitative objectives, Acta Inform. 58 (6) (2021) 585–610.
[26] N. Bulling, V. Goranko, Combining quantitative and qualitative reasoning in concurrent multi-player games, Auton. Agents Multi-Agent Syst. 36 (1) (2022) 1–33.
[27] X. Liao, Maximum satisfiability approach to game theory and network security, Ph.D. thesis, Kyushu University, 2014.
[28] R. Dimitrova, M. Ghasemi, U. Topcu, Reactive synthesis with maximum realizability of linear temporal logic specifications, Acta Inform. 57 (1) (2020) 107–135.
[29] B. Finkbeiner, C. Hahn, H. Torfah, Model checking quantitative hyperproperties, in: International Conference on Computer Aided Verification, Springer, 2018, pp. 144–163.
[30] D. Kroening, J. Ouaknine, O. Strichman, T. Wahl, J. Worrell, Linear completeness thresholds for bounded model checking, in: International Conference on Computer Aided Verification, Springer, 2011, pp. 557–572.
[31] F. Bacchus, J. Berg, M. Järvisalo, R. Martins, A. Niskanen (Eds.), MaxSAT Evaluation 2022: Solver and Benchmark Descriptions, Department of Computer Science Series of Publications B, Department of Computer Science, University of Helsinki, Finland, 2022.
[32] J.F. Kurose, R. Simha, A microeconomic approach to optimal resource allocation in distributed computer systems, IEEE Trans. Comput. 38 (5) (1989) 705–717.
[33] A. Mukherjee, P. Goswami, Z. Yan, L. Yang, J.J. Rodrigues, Adai and adaptive pso-based resource allocation for wireless sensor networks, IEEE Access 7 (2019) 131163–131171.
[34] F. Chang, J. Ren, R. Viswanathan, Optimal resource allocation in clouds, in: 2010 IEEE 3rd International Conference on Cloud Computing, IEEE, 2010, pp. 418–425.
[35] E. Semsar-Kazerooni, K. Khorasani, Multi-agent team cooperation: a game theory approach, Automatica 45 (10) (2009) 2205–2213.
[36] D. Fisman, O. Kupferman, Y. Lustig, Rational synthesis, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2010, pp. 190–204.
[37] W. Jamroga, W. Penczek, T. Sidoruk, P. Dembiński, A. Mazurkiewicz, Towards partial order reductions for strategic ability, J. Artif. Intell. Res. 68 (2020) 817–850.
[38] M. Cohen, M. Dam, A. Lomuscio, H. Qu, A symmetry reduction technique for model checking temporal-epistemic logic, in: Twenty-First International Joint Conference on Artificial Intelligence, Citeseer, 2009.
[39] N. Timm, S. Gruner, M. Nxumalo, J. Botha, Model checking safety and liveness via k-induction and witness refinement with constraint generation, Sci. Comput. Program. 200 (2020) 102532.