State of the Art in Logics for Verification of Resource-Bounded Multi-Agent Systems

Natasha Alechina[1] and Brian Logan[2]

[1] Utrecht University
Utrecht, The Netherlands
n.a.alechina@uu.nl
[2] University of Nottingham
Nottingham, UK
bsl@cs.nott.ac.uk

Abstract. Approaches to the verification of multi-agent systems are typically based on games or transition systems defined in terms of states and actions. However such approaches often ignore a key aspect of multi-agent systems, namely that the agents' actions require (and sometimes produce) resources. We survey previous work on the verification of multi-agent systems that takes resources into account, extending substantially a survey from 2016 [9].

1 Introduction

A multi-agent system (MAS) is a system that is composed of multiple interacting agents. An agent is an *autonomous* entity that has the ability to collect information, reason about it, and perform actions in pursuit of its goals or on behalf of others. Examples of agents are controllers for satellites, non-driver transport systems such as UAVs, smart manufacturing cells, smart energy grids, and nodes in sensor networks.

Many distributed hardware and software systems can be naturally modelled as multi-agent systems. Such systems are often extremely complex, and the interaction between the components and their environment can lead to undesired behaviours that are difficult to predict in advance. With the increasing use of autonomous agents in safety critical systems, there is a growing need to *verify* that their behaviour conforms to the desired system specification, and over the last decade verification of multi-agent systems has become a thriving research area [35].

A key approach to the verification of MAS is *model checking*. Model checking involves checking whether a model of the system satisfies a temporal logic formula corresponding to some aspect of the system specification. In a model-checking approach to the verification of multi-agent systems, a MAS is represented by a finite state transition system.[3] A state transition system consists of a set of states and transitions between them. Intuitively, each state of a MAS corresponds to a tuple of states of the agents and of the environment, and each transition corresponds to actions performed by the agents. Each state is labelled with atomic propositions that are true in that state. A

[3] There is work on model-checking infinite state transition systems, see, for example, [18], but in this paper we concentrate on the finite case.

standard assumption is that each state in the system has at least one outgoing transition (if a state is a deadlock state in the original MAS, we can model this by adding a transition to itself by some null action and labelling it with a 'deadlock' proposition). Properties of the system to be verified are expressed in an appropriate temporal logic L. The *model-checking problem for* L is, given a state transition system M (and possibly a state s) and an L formula ϕ, check whether ϕ is true in M (at state s, or on all paths from s, etc.). For example, Linear Time Temporal Logic (LTL) can express properties of infinite runs through the system using a unary operator 'in the next state' $\bigcirc$ ($\bigcirc\phi$ means that on this path, the next state satisfies ϕ) and a binary operator 'until' $\mathcal{U}$ ($\phi\mathcal{U}\psi$ means that on this path, ψ holds after finitely many steps, and before that, ϕ holds in every state). Using these operators, one can define operators such as $\Diamond$ (in some state on this path) and $\Box$ (in every state on this path) and specify properties of interest of the system, such as deadlock never happens ($\Box\neg d$) or every request is eventually answered ($\Box(r \rightarrow \Diamond a)$). In model checking MAS, such temporal logics are often extended with additional modalities capturing the knowledge of agents, or the strategic ability of groups of agents. Model checking has the advantage that it is a fully automated technique, which facilitates its use in the MAS development process.[4] A wide range of approaches to model-checking MAS have been proposed in the literature, ranging from the adaptation of standard model-checking tools, e.g., [20, 21] to the development of special-purpose model checkers for multi-agent systems, e.g., [41, 33].

In many multi-agent systems, agents are *resource-bounded*, in the sense that they require resources in order to act. Actions require time to complete and typically require additional resources depending on the application domain, for example energy or money. For many applications, the availability or otherwise of resources is critical to the properties to be verified: a multi-agent system may have very different behaviours depending on the resource endowment of the agents that comprise it. For example, an agent with insufficient energy may be unable to complete a task in the time assumed by a team plan if it has to recharge its battery before performing the task.

In this paper we survey state of the art in the emerging field of logics for verification of resource-bounded agents, and highlight a number of challenges that must be overcome to allow practical verification of resource-bounded MAS. We argue that recent work on the complexity of model-checking for logics of strategic ability with resources offers the possibility of significant progress in the field, new verification approaches and tools, and the ability to verify the properties of a large, important class of autonomous system that were previously out of reach.

The remainder of the paper is organised as follows. In Section 2, we introduce some necessary background material on weighted games. Reachability in weighted games can be seen as a verification technique in its own right; however, it is included here as a source of technical results relevant for strategic resource logics. In this section, we also introduce the syntax and semantics of strategy logics (without resources) that are the underlying formalism for resource logics. In Section 3 we briefly survey recent work in resource logics and study two logics, RB$\pm$ATL and RB$\pm$ATL*, in greater detail. We conclude in Section 4 with a summary of results and open problems.

[4] Another strand of work focusses on theorem proving, e.g., [44], but such approaches typically require user interaction to guide the search for a proof.

2 Background

In this section, we recall relevant definitions and results for energy games, vector addition systems with states, and the logics of strategic ability ATL and ATL*.

We first introduce some notational conventions. In what follows, we use the usual point-wise notation for vector comparison and addition. In particular, $(b_1, \ldots, b_n) \leq (d_1, \ldots, d_n)$ iff $b_i \leq d_i \; \forall i \in \{1, \ldots, n\}$, $(b_1, \ldots, b_n) = (d_1, \ldots, d_n)$ iff $b_i = d_i \; \forall \; i \in \{1, \ldots, n\}$, and $(b_1, \ldots, b_n) + (d_1, \ldots, d_n) = (b_1 + d_1, \ldots, b_n + d_n)$ and $(b_1, \ldots, b_n) - (d_1, \ldots, d_n) = (b_1 - d_1, \ldots, b_n - d_n)$. We define $(b_1, \ldots, b_n) < (d_1, \ldots, d_n)$ as $(b_1, \ldots, b_n) \leq (d_1, \ldots, d_n)$ and $(b_1, \ldots, b_n) \neq (d_1, \ldots, d_n)$. Given a function f returning a vector, we denote by f_i the function that returns the i-th component of the vector returned by f. We use bold letters to denote vectors.

Given a set S, the set of finite sequences of elements from S is denoted by S^+. For a sequence $\lambda = s_1 \ldots s_k \in S^+$, we use the notation $\lambda[i] = s_i$ for $i \leq k$, $\lambda[i, j] = s_i \ldots s_j$ $\forall \; 1 \leq i \leq j \leq k$, and $|\lambda| = k$ for the length of λ.

2.1 Energy Games and Vector Addition Systems with States

Distributed systems that produce and consume resources have been modelled using a variety of approaches, including Petri nets, energy games and vector addition systems with states. In this section, we briefly recall some results from these areas relevant to resource logics and model checking resource-bounded MAS. We will first briefly introduce a version of *energy games* before introducing a variant of *alternating vector addition systems with states (AVASS)*. We focus on the reachability and non-termination problems for AVASS, as these are the most relevant for the results on resource logics in Section 3.

Energy Games *Energy games* [28] are games between two players, played on *multi-weighted game graphs*.

Definition 1. *A multi-weighted game graph of dimension r is a tuple (S, r, R) where S is the set of vertices, $R \subseteq S \times \mathbb{Z}^r \times S$ is a finite set of edges labelled by a vector of integers of length r called a weight. Each vertex has at least one outgoing edge. The set of vertices is partitioned into two sets, Player 1 vertices S_1 and Player 2 vertices S_2.*

The dimension is the number of resource types, where resource types can be, e.g., energy, memory or some other kind of capacity, time, money, etc. The vertices can be thought of as states, and edges as transitions between states with associated costs and rewards for each resource type. The weight of an edge describes how the corresponding transition affects the resource amounts. Note that, in the graph, there are no resource vectors associated with the vertices, so that the structure can be finitely represented. However we can talk about *configurations* which are pairs $(s, \mathbf{v})$ where s is a vertex and $\mathbf{v}$ a vector of resources: intuitively, $\mathbf{v}$ is the resource amounts available in s in this configuration. A *path* is a finite sequence of configurations $(s_1, \mathbf{v}_1), \ldots, (s_n, \mathbf{v}_n)$, such that for each j with $1 \leq j \leq n$ there is an edge $(s_j, \mathbf{v}_{j+1} - \mathbf{v}_j, s_{j+1})$. A *play from vertex s* is an infinite sequence of configurations $\rho = (s, \mathbf{v}), \ldots,$ such that every finite prefix is a path. A *strategy* for a player i is a function F_i taking as input a path $\rho \cdot (s, \mathbf{v})$ ending in Player i vertex s and returning an edge $F_i(\rho \cdot (s, \mathbf{v}))$ of the form $(s, \mathbf{u}, s')$ from E. A play $\rho = (s_1, \mathbf{v}_1), \ldots, (s_j, \mathbf{v}_j) \ldots$ is *consistent* with a strategy F_p for Player p if whenever s_j is in S_p, then $F_p(\rho[1, j]) = (s_j, \mathbf{v}_{j+1} - \mathbf{v}_j, s_{j+1})$.

Definition 2. *Given a multi-weighted graph (S, r, R), an initial vertex s, and a vector $\mathbf{b} \in \mathbb{N}^r$, a play ρ from s is* winning *for Player 1 in the energy game on (S, r, R) with initial credit $\mathbf{b}$ if for all configurations $\rho[j] = (s_j, \mathbf{v}_j)$, $\mathbf{v}_j \geq \mathbf{0}$. Otherwise, Player 2 wins the play. Player 1 wins the energy game on (S, r, R) from s with initial credit $\mathbf{b}$ if there exists a winning strategy F_1 for Player 1, that is, a strategy such that for all strategies F_2 of Player 2, the play consistent with both strategies is winning for Player 1.*

Intuitively, starting in state s with initial credit (resource allocation) $\mathbf{b}$, Player 1 can play forever without any resource amount dropping below 0. Clearly, the higher the initial credit, the better for Player 1; if Player 1 has a winning strategy for $(s, \mathbf{b})$, and $\mathbf{b} \leq \mathbf{b}'$, then Player 1 has a winning strategy from $(s, \mathbf{b}')$.

Definition 3. *The following problem is the existence of a winning strategy for Player 1 with known initial credit.*

Input: *A multi-weighted graph (S, R, r), an initial state $s \in S$ and an initial credit $\mathbf{b}$.*
Question: *Does Player 1 have a winning strategy in the corresponding energy game?*

An *energy game with unknown initial credit* starting in s is won by Player 1 iff for *some* initial credit, Player 1 has a winning strategy.

Definition 4. *The following problem is the existence of a winning strategy for Player 1 with unknown initial credit.*

Input: *A multi-weighted graph (S, R, r) and an initial state $s \in S$.*
Question: *Does Player 1 have a winning strategy in the corresponding energy game for* some *initial credit $\mathbf{b}$?*

Both problems (existence of a winning strategy for known and unknown initial credit) were first shown to be decidable in [22]. In [37] both problems were shown to be decidable in 2EXPTIME (polynomial in the size of the graph, double exponential in the dimension r). In [37] it was also shown that the set of all Pareto optimal (non-dominated) initial credits for which Player 1 has a winning strategy is computable in time doubly exponential in the dimension and pseudo-polynomial in the number of states and edges.

There are many versions of energy games: with only unit costs, with only one resource type, with imperfect information. A version with finite strategies was studied in [28] and shown to be decidable and in coNP.

Alternating Vector Addition Systems with State An *alternating vector addition system with state* (AVASS) can be used as a setting for various two player games. There are many different versions of AVASS and decision problems for them. The game semantics for AVASS presented below was introduced in [38].

Definition 5. *An* alternating vector addition system with states *(AVASS) is a tuple $A = (S, r, R_1, R_2)$, where S is a finite set of states, r is the dimension (number of resource types), $R_1 \subseteq S \times \mathbb{Z}^r \times S$ and $R_2 \subseteq S^3$.*

Intuitively, R_1 edges correspond to Player 1 moves, and R_2 triples (s, s_1, s_2) correspond to Player 2 choices of where to move from the state s, to s_1 or to s_2. Note that unlike in energy games, the setting is *asymmetric* in that only Player 1 moves change resource amounts. A path of configurations is defined the same way as for energy games: in a configuration $(s, \mathbf{b})$, if the next move is $(s, \mathbf{v}, s') \in R_1$, then the next configuration is $(s', \mathbf{b} + \mathbf{v})$; if the next move is $(s, s_1, s_2) \in R_2$, then, depending on the choice made by Player 2, the next configuration is either $(s_1, \mathbf{b})$ or $(s_2, \mathbf{b})$.

The following problem is essentially the same as the existence of a winning strategy for Player 1 in an energy game with known initial credit:

Definition 6. *The following problem is the known initial credit* non-termination *problem for AVASS:*

Input: *An AVASS $A = (S, r, R_1, R_2)$, an initial state $s \in S$ and an initial credit* $\mathbf{b}$.
Question: *Does Player 1 have a strategy such that every play consistent with this strategy is infinite and all resource amounts in configurations on the path are non-negative?*

This problem was shown to be decidable and in $(r-1)$-EXPTIME in [22], 2EXPTIME hard in [30], and in 2EXPTIME in [37]. The unknown initial credit version of the problem is also 2EXPTIME-complete [37]. The set of all Pareto optimal initial credits for which Player 1 has a winning strategy can be computed in 2EXPTIME [37].

Another problem which has been studied in the AVASS literature is state reachability. The state reachability problem is whether Player 1 has a strategy to reach a particular state while ensuring resource amounts remain non-negative (as opposed to reachability of a particular configuration $(s', \mathbf{v})$, which is undecidable, [40]). The state reachability problem for energy games is undecidable [2].

Definition 7. *The following problem is the known initial credit* state reachability *problem for AVASS:*

Input: *An AVASS $A = (S, r, R_1, R_2)$, an initial state $s \in S$, an initial credit* $\mathbf{b}$ *and state $s' \in S$.*
Question: *Does Player 1 have a strategy such that every path generated by this strategy eventually reaches a configuration where the state is s', and until that configuration, all resource amounts on the path are non-negative?*

This problem was shown to be decidable in [43], and to be 2EXPTIME-complete in [30]. In the same paper, the state reachability problem with unknown initial credit was also shown to be 2EXPTIME-complete. The set of all Pareto optimal initial credits for which Player 1 has a winning strategy can be computed in 2EXPTIME [37].

Parity Games on AVASS Another kind of games on AVASS is *parity games*. Let $A = (S, r, R_1, R_2)$ be an AVASS. A colouring col is defined as a map $S \to \{0, \dots, k\}$ for some $k \geq 1$.

Definition 8. *The* parity game problem for AVASS *is as follows:*

Input: *An AVASS A, an initial state $s \in A$, an initial credit $\mathbf{b} \in \mathbb{N}^r$ and a colouring $col : S \to \{0, \dots, k\}$*

Question: *Does Player 1 have a strategy in $(s, \mathbf{b})$ such that every play consistent with this strategy is infinite, resource amounts in configurations on the path are non-negative, and on every play the maximal colour that appears infinitely often is even?*

The parity game problem for alternating VASS is decidable. This was shown in [5] to be a consequence of Corollary 2 in [1] which states the decidability of parity games for single-sided VASS. A single-sided VASS is an AVASS where the set of states is partitioned into S_1 and S_2, R_1 transitions start from states in S_1, R_2 transitions start from states in S_2, and there is at most one R_2 transition from each S_2 state.

2.2 Strategy Logics

In this section, we briefly recall some key results for the strategy logics Alternating Time Temporal Logic (ATL) [16] and the more expressive ATL* that are the underlying formalisms for many of the resource logics discussed in Section 3.

Alternating Time Temporal Logic ATL generalises other temporal logics such as Computation Tree Logic (CTL) [29] (which can be seen as a one-agent ATL) by introducing a notion of strategic ability. ATL allows us to express properties relating to the strategic abilities of a coalition or set of agents regardless of what the other agents in the system do.

ATL is interpreted over concurrent game structures. A concurrent game structure is a transition system in which edges correspond to a tuple of actions performed simultaneously by all the agents (see below and Figure 1 for an example).

Definition 9. *A concurrent game structure (CGS) is a tuple $M = (Agt, S, \Pi, \pi, Act, d, \delta)$ where:*

- *Agt is a non-empty finite set of n agents,*
- *S is a non-empty finite set of states;*
- *Π is a finite set of propositional variables and $\pi : \Pi \to \wp(S)$ is a truth assignment which associates each proposition in Π with a subset of states where it is true;*
- *Act is a non-empty set of actions*
- *$d : S \times Agt \to \wp(Act) \setminus \{\emptyset\}$ is a function which assigns to each $s \in S$ a non-empty set of actions available to each agent $a \in Agt$. We denote joint actions by all agents in Agt available at s by $D(s) = d(s, a_1) \times \cdots \times d(s, a_n)$;*
- *$\delta : S \times Act^{|Agt|} \to S$ is a partial function that maps every $s \in S$ and joint action $\sigma \in D(s)$ to a state resulting from executing σ in s.*

Given a CGS M and a state $s \in S$, a *joint action by a coalition* $A \subseteq Agt$ is a tuple $\sigma = (\sigma_a)_{a \in A}$ (where σ_a is the action that agent a executes as part of σ, the ath component of σ) such that $\sigma_a \in d(s, a)$. The set of all joint actions for A at state s is denoted by $D_A(s)$.

Given a joint action by Agt $\sigma \in D(s)$, σ_A (a projection of σ on A) denotes the joint action executed by A as part of σ: $\sigma_A = (\sigma_a)_{a \in A}$. The set of all possible outcomes of a joint action $\sigma \in D_A(s)$ at state s is:

$$out(s, \sigma) = \{s' \in S \mid \exists \sigma' \in D(s) : \sigma = \sigma'_A \wedge s' = \delta(s, \sigma')\}$$

Depending on the variant of ATL, a strategy is a choice of actions which either only depends on the current state (memoryless strategy) or on the finite history of the current state (perfect recall strategy). In this survey, we concentrate mainly on perfect recall strategies. A *strategy for a coalition* $A \subseteq Agt$ in a CGS M is a mapping $F_A : S^+ \to Act^{|A|}$ such that, for every $\lambda \in S^+$, $F_A(\lambda) \in D_A(\lambda[|\lambda|])$. A computation (infinite path) λ is consistent with a strategy F_A iff, for all i, $\lambda[i+1] \in out(\lambda[i], F_A(\lambda[1, i]))$. We denote by $out(s, F_A)$ the set of all computations λ starting from s that are consistent with F_A.

The language of ATL contains atomic propositions, boolean connectives $\neg, \wedge$, etc. and modalities $\langle\langle A \rangle\rangle \bigcirc$, $\langle\langle A \rangle\rangle \Box$ and $\langle\langle A \rangle\rangle \mathcal{U}$ for each subset A of the set of all agents Agt (or coalition, in ATL terms), which express the strategic ability of the coalition A. $\langle\langle A \rangle\rangle \bigcirc \phi$ means that the coalition of agents A has a choice of actions such that, regardless of what the other agents in the system do, ϕ will hold in the next state. $\langle\langle A \rangle\rangle \Box \phi$ means that coalition A has a strategy to keep ϕ true forever, regardless of what the other agents do. Finally, $\langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ means that A has a strategy to ensure that after finitely many steps ψ holds, and in all the states before that, ϕ holds.

Given a CGS M and a state s of M, the truth of an ATL formula ϕ with respect to M and s is defined inductively on the structure of ϕ as follows:

- $M, s \models p$ iff $s \in \pi(p)$;
- $M, s \models \neg \phi$ iff $M, s \not\models \phi$;
- $M, s \models \phi \vee \psi$ iff $M, s \models \phi$ or $M, s \models \psi$;
- $M, s \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff $\exists$ strategy F_A such that for all $\lambda \in out(s, F_A)$, $M, \lambda[2] \models \phi$;
- $M, s \models \langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ iff $\exists$ strategy F_A such that for all $\lambda \in out(s, F_A)$, $\exists i$ such that $M, \lambda[i] \models \psi$ and $M, \lambda[j] \models \phi$ for all $j \in \{1, \ldots, i-1\}$;
- $M, s \models \langle\langle A \rangle\rangle \Box \phi$ iff $\exists$ strategy F_A such that for all $\lambda \in out(s, F_A)$, for all i, $M, \lambda[i] \models \phi$.

Example Figure 1 illustrates a simple ATL model of a system with two agents, 1 and 2, and actions α, β, γ and $idle$. Action tuples on the edges show the actions of each agent, for example, in the transition from state s_I to s, agent 1 performs action α and agent 2 performs $idle$. In this system, in state s_I, agent 1 has a (memoryless) strategy to enforce that p holds eventually in the future no matter what agent 2 does, which can be expressed in ATL as $\langle\langle \{1\} \rangle\rangle \top \mathcal{U} p$. Similarly, in s_I agent 1 has a memoryless strategy to keep $\neg p$ true forever, so $\langle\langle \{1\} \rangle\rangle \Box \neg p$ holds in s_I.

Definition 10. *The following problem is the model checking problem for ATL:*

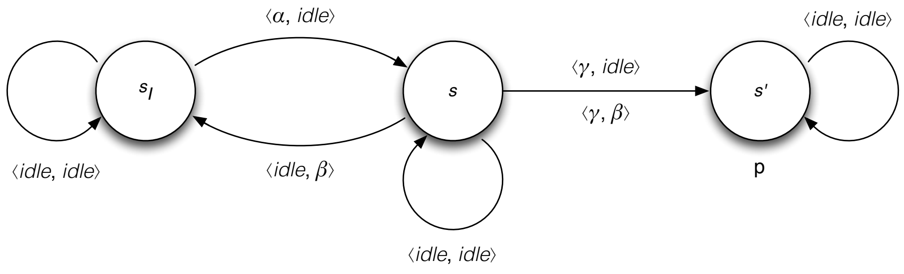

Fig. 1: Example of a state transition system.

Input: *A CGS M, a formula ϕ of ATL, and a state $s \in M$.*
Question: *Does it hold that $M, s \models \phi$?*

The model-checking problem for ATL can be solved in time polynomial in the size of the transition system and the property [16], and there exist model-checking tools for ATL, for example, MOCHA [17] and MCMAS [41].

ATL* ATL* is strictly more expressive than ATL in allowing arbitrary combinations of temporal modalities and booleans after the coalition modalities. The syntax of ATL* includes two kinds of formulas, state formulas ϕ and path formulas γ. Formulas of ATL* are defined by the following syntax:

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \langle\langle A \rangle\rangle \gamma$$

$$\gamma ::= \phi \mid \neg\gamma \mid \gamma \vee \gamma \mid \bigcirc\gamma \mid \gamma\,\mathcal{U}\,\gamma \mid \Box\gamma$$

where $p \in \Pi$ is a proposition and $A \subseteq Agt$.

The language of ATL* is interpreted on the same CGS as ATL. However, there are two satisfaction relations, $\models_s$ for state formulas, and $\models_p$ for path formulas:

– $M, s \models_s p$ iff $s \in \pi(p)$;
– $M, s \models_s \neg\phi$ iff $M, s \not\models_s \phi$;
– $M, s \models_s \phi \vee \psi$ iff $M, s \models_s \phi$ or $M, s \models_s \psi$;
– $M, s \models_s \langle\langle A^{\boldsymbol{b}} \rangle\rangle \gamma$ iff exists a strategy F_A such that for all $\lambda \in out(s, F_A)$, $M, \lambda \models_p \gamma$;
  $M, \lambda \models_p \phi$ iff $M, \lambda[1] \models_s \phi$ (for state formulas ϕ)
– $M, \lambda \models_p \bigcirc\gamma$ iff $M, \lambda[2, \infty) \models_p \gamma$
– $M, \lambda \models_p \gamma_1 \mathcal{U} \gamma_2$ iff $\exists k$ such that $M, \lambda[k, \infty) \models \gamma_2$ and $M, \lambda[j, \infty) \models \gamma_1$ for all $j \in \{1, \ldots, k-1\}$.
– $M, \lambda \models_p \Box\gamma$ iff for all j $M, \lambda[j, \infty) \models_p \gamma$.

Definition 11. *The following problem is the model checking problem for ATL*:*

Input: *A CGS M, a state formula ϕ of ATL*, and a state s in M.*
Question: *Does it hold that $M, s \models_s \phi$?*

The complexity of the model checking problem for ATL* is 2EXPTIME-complete [16].

3 Resource Logics

In order to model multi-agent systems where the actions of agents produce and consume resources, it is necessary to modify strategy logics in two ways. The first modification is to add resource annotations to the actions in the transition system: for each individual action and each resource type, we need to specify how many units of this resource type the action produces or consumes. For example, suppose that there are two resource types, r_1 and r_2 (e.g., energy and money). Then we can specify that action α in Figure 1 produces two units of r_1 and consumes one unit of r_2, action β consumes one unit of r_1 and produces one unit of r_2, action γ consumes five units of r_1, and action $idle$ does not produce or consume any resources. Clearly, this makes the transition system of a CGS resemble multi-weighted graphs or AVASS introduced in Section 2.1.

The second modification is to extend the logical language so that we can express properties related to resources. For example, we may want to express a property that a group of agents A can eventually reach a state satisfying ϕ or can maintain the truth of ψ forever, provided that they have available n_1 units of resource type r_1 and n_2 units of resource type r_2. Such statements about coalitional ability under resource bounds can be expressed in an extension of ATL where coalitional modalities are annotated with a resource bound on the strategies available to the coalition. We call logics where every action is associated the resources it produces and/or consumes and where the syntax allows the resource requirements of agents to be expressed, *resource logics*.

To illustrate the properties resource logics allow us to express, consider the model in Figure 1 with the production and consumption of resources by actions specified above. In this setting, we can verify if agent 1 can eventually enforce p provided that it has one unit of r_2 in state s_I, or whether the coalition of agents $\{1, 2\}$ can achieve p under this resource bound by working together. There are surprisingly many different ways of measuring costs of strategies and deciding which actions are executable by the agents given the resources available to them, but under at least one possible semantics, the answer to the first question is no and to the second one yes, but the latter requires a perfect recall strategy (the two agents should loop between states s_I and s until they produce a sufficient amount of resource r_1, and then execute actions corresponding to the $\langle \gamma, idle \rangle$ transition from s to s').

Clearly, the model-checking problem for temporal logics is a special case of the model-checking problem for the corresponding resource logics. The question is, how much harder does the model-checking problem become when resources are added?

3.1 Overview of Resource Logics

In this section, we briefly review the historical development of resource logics, and introduce some resource logics in more detail. We focus on expressiveness and model-checking complexity, as these features determine the suitability of a particular logic for practical verification.

Consumption of Resources Early work on resource logics considered only consumption of resources (i.e., no action produces resources), and initial results were encouraging.

One of the first logics capable of expressing resource requirements of agents was a version of Coalition Logic (CL),[5] called Resource-Bounded Coalition Logic (RBCL), where actions only consume (and don't produce) resources. RBCL was introduced in [3] with the primary motivation of modelling systems of resource-bounded reasoners (with three resource types: time, space, and communication cost), however the framework is sufficiently general to model any kind of action. The model-checking problem for this logic was shown to be decidable in [11] in polynomial time in the transition system and the property, and exponential in the number of resource types.

A resource-bounded version of ATL, RB-ATL, where again actions only consume (and not produce) resources was introduced in [4]. It was also shown that the model-checking problem for this logic is decidable in time polynomial in the size of the transition system and exponential in the number of resource types. (For a single resource type, e.g., energy, the model-checking problem is no harder than for ATL.) Its syntax is the same as RB$\pm$ATL given in Section 3.2 below, but in the semantics no actions produce resources. Probabilistic RB-ATL was introduced in [42] and its model checking problem shown to be decidable in EXPTIME.

Practical work on model-checking standard computer science transition systems (not multi-agent systems) with resources also falls in the category of consumption-only systems, for example the probabilistic model-checking of systems with numerical resources in the PRISM model-checker [39] assumes costs monotonically increasing with time.

Bounded Production and Undecidability in the Unbounded Setting However, when resource production is considered in addition to consumption, the situation changes. In a separate strand of work, a range of different formalisms for reasoning about resources was introduced in [25, 23]. In those formalisms, both consumption and production of resources was considered. In [24] it was shown that the problem of halting on empty input for two-counter automata [36] can be reduced to the model-checking problem for several of their resource logics. Since the halting problem for two-counter automata is undecidable, the model-checking problem for a variety of resource logic with production of resources is undecidable. The reduction uses two resource types (to represent the values of the two counters) and either one or two agents depending on the version of the logic (whether the agents have perfect recall, whether the formula talking about coalition A can also specify resource availability for remaining agents, and whether nested operators 'remember' initial allocation of resources or can be evaluated independently of such initial allocation).

[5] CL is a fragment of ATL with only the next time $\langle\langle A\rangle\rangle\bigcirc$ modality.

The only decidable cases considered in [23] are an extension of CTL with resources (essentially one-agent ATL) and a version where on every path only a fixed finite amount of resources can be produced. In [23], the models satisfying this property are called bounded, and the authors note that RBCL and RB-ATL are logics over a special kind of bounded models (where no resources are produced at all). Other decidability results for bounded resource logics have also been reported in the literature. For example, [31] define a decidable logic, PRB-ATL (Priced Resource-Bounded ATL), where the total amount of resources in the system has a fixed bound. The model-checking algorithm for PRB-ATL requires time polynomial in the size of the model and exponential in the number of resource types and the resource bound on the system. In [32] an EXPTIME lower bound in the number of resource types for the PRB-ATL model-checking problem is shown.

A general logic over systems with numerical constraints called QATL* was introduced in [26]. In that paper, more undecidability results for the model-checking problem of QATL* and its fragments were shown. For example, QATL (Quantitative ATL) is undecidable even if no nestings of coalition modalities is allowed. The main proposals for restoring decidability to the model-checking problem for QATL in [26] are removing negative payoffs (similar to removing resource production) and also introducing memoryless strategies. Shared resources were considered in [27]; most of the cases considered there have undecidable model-checking (apart from the case of a single shared resource, which has decidable model-checking).

In summary, one approach to decidable model checking in the presence of resource production is to bound the amount of resources produced globally in the model. For some systems of resource-bounded agents, this is a reasonable restriction. For example, agents that need energy to function and are able to charge their battery, can never 'produce' more energy than the capacity of their battery. This is a typical bounded system. A special case of bounded systems, where model checking is even more tractable, are systems where one of the resources is always consumed by any action. A typical example of such a resource is time. Several resource logics with *diminishing resource* were investigated in [10] and shown to have a PSPACE or EXPSPACE model checking procedure (while the corresponding logic without diminishing resource sometimes has undecidable model checking).

In the next couple of sections, we report results for resource logics with unbounded production of resources and a decidable model checking problem.

3.2 RB$\pm$ATL

In [12] a version of ATL, RB$\pm$ATL, was introduced where actions both produce and consume resources. The models of the logic do not impose bounds on the overall production of resources, and the agents have perfect recall. The syntax of RB$\pm$ATL is very similar to that of ATL, but coalition modalities have superscripts which represent resource allocation to agents. Instead of stating the existence of *some* strategy, they state the existence of a strategy such that every computation generated by following this strategy consumes at most the given amount of resources. Coming back to the example, the property that agent 1 can eventually enforce p provided that it has one unit of r_2 can be expressed as $\langle\langle \{1\}^{(0,1)} \rangle\rangle \top \, \mathcal{U} \, p$. Here, $(0, 1)$ is the allocation of 0 units of r_1

and 1 unit of r_2 to coalition $\{1\}$. In RB$\pm$ATL, resource allocation is only shown for the proponent agents, $\{1\}$ in this case. Versions of resource logic where opponents are also resource-bounded all have an undecidable model-checking problem, see [23]. It is also possible to consider individual allocations of resources to agents in the proponent coalition, which would affect complexity results below for one resource type.

Formally, the syntax of RB$\pm$ATL is defined relative to the following sets: $Agt = \{a_1, \ldots, a_n\}$ is a set of n agents, $Res = \{res_1, \ldots, res_r\}$ is a set of r resource types, Π is a set of propositions, and $\mathcal{B} = \mathbb{N}^r$ is a set of resource bounds. Formulas of RB$\pm$ATL are defined by the following syntax:

$$\phi, \psi ::= p \mid \neg\phi \mid \phi \vee \psi \mid \langle\langle A^{\boldsymbol{b}} \rangle\rangle \bigcirc \phi \mid \langle\langle A^{\boldsymbol{b}} \rangle\rangle \phi \mathcal{U} \psi \mid \langle\langle A^{\boldsymbol{b}} \rangle\rangle \Box \phi$$

where $p \in \Pi$ is a proposition, $A \subseteq Agt$, and $\boldsymbol{b} \in \mathcal{B}$ is a resource bound. Here, $\langle\langle A^{\boldsymbol{b}} \rangle\rangle \bigcirc \phi$ means that a coalition A can ensure that the next state satisfies ϕ under resource bound $\boldsymbol{b}$. $\langle\langle A^{\boldsymbol{b}} \rangle\rangle \phi \mathcal{U} \psi$ means that A has a strategy to enforce ψ while maintaining the truth of ϕ, and the cost of this strategy is at most $\boldsymbol{b}$. Finally, $\langle\langle A^{\boldsymbol{b}} \rangle\rangle \Box \phi$ means that A has a strategy to maintain ψ forever, and the cost of this strategy is at most $\boldsymbol{b}$.

The language is interpreted on resource-bounded concurrent game structures.

Definition 12. *A resource-bounded concurrent game structure (RB-CGS) is a tuple $M = (Agt, Res, S, \Pi, \pi, Act, d, c, \delta)$ where:*

- *$Agt, S, \Pi, \pi, Act, d, \delta$ are as in Definition 9;*
- *Res is a non-empty finite set of r resource types,*
- *$c : S \times Act \to \mathbb{Z}^r$ is a partial function which maps a state s and an action σ to a vector of integers, where the integer in position i indicates consumption or production of resource r_i by the action (here, we assume negative value for consumption and positive value for production for consistency with AVASS, unlike in [12]).*

A strategy for a set of agents A is a function $F_A : S^+ \to Act^A$ such that $F_A(\lambda) \in D_A([\lambda[|\lambda|])$. Given a bound $\boldsymbol{b} \in \mathcal{B}$, a computation $\lambda \in out(s, F_A)$ is $\boldsymbol{b}$-consistent iff for every i,

$$\boldsymbol{b} + \Sigma^i c(F_A(\lambda[1, i])) \geq \mathbf{0}$$

In other words, if agents start with allocation $\boldsymbol{b}$, the amount of resources any of the agents have on the computation is never negative for any resource type.

A strategy F_A is $\boldsymbol{b}$-consistent in s, if all computations in $out(s, F_A)$ are $\boldsymbol{b}$-consistent.

Given a RB-CGS M and a state s of M, the truth of an RB$\pm$ATL formula ϕ with respect to M and s is defined inductively on the structure of ϕ as follows:

- $M, s \models p$ iff $s \in \pi(p)$;
- $M, s \models \neg\phi$ iff $M, s \not\models \phi$;
- $M, s \models \phi \vee \psi$ iff $M, s \models \phi$ or $M, s \models \psi$;
- $M, s \models \langle\langle A^{\boldsymbol{b}} \rangle\rangle \bigcirc \phi$ iff $\exists$ $\boldsymbol{b}$-consistent strategy F_A such that for all $\lambda \in out(s, F_A)$, $M, \lambda[2] \models \phi$;
- $M, s \models \langle\langle A^{\boldsymbol{b}} \rangle\rangle \phi \mathcal{U} \psi$ iff $\exists$ $\boldsymbol{b}$-consistent strategy F_A such that for all $\lambda \in out(s, F_A)$, $\exists i$ such that $M, \lambda[i] \models \psi$ and $M, \lambda[j] \models \phi$ for all $j \in \{1, \ldots, i-1\}$.

– $M, s \models \langle\langle A^{\boldsymbol{b}} \rangle\rangle \Box \phi$ iff $\exists$ $\boldsymbol{b}$-conststent strategy F_A such that for all $\lambda \in out(s, F_A)$, for all i, $M, \lambda[i] \models \phi$.

Definition 13. *The following problem is the model checking problem for RB±ATL:*

Input: *A RB-CGS M, a formula ϕ of RB±ATL, and a state $s \in M$.*
Question: *Does it hold that $M, s \models \phi$?*

The model-checking problem for RB±ATL is decidable. The existence of a decidable resource logic with unbounded production was surprising, as it was the first indication that it is possible to automatically verify properties of this important class of resource-bounded multi-agent systems. In [12], decidability of the model-checking problem was shown by producing a direct model checking algorithm and arguing that it terminates due to the fact that in any sequence of elements from $\mathbb{N}^r$, eventually two elements are comparable in $\leq$ (well-quasi ordering of $\mathbb{N}^r$).

3.3 Correspondence between Games on AVASS and RB±ATL Semantics

There are clear similarities between RB±ATL semantics and decidable problems for AVASS and energy games. In [5] these similarities were made precise, and the model checking problem for RB±ATL was shown to be polynomial in the size of the model and the formula, and double exponential in the number of resource types, by reducing the model checking to decision problems on AVASS. We will briefly recapitulate the correspondence here.

For the purposes of making the correspondence easier to state, the definitions of AVASS and the state reachability problem were generalised as follows, without affecting the complexity of decision problems ([5], Lemma 7):

– instead of $R_2 \subseteq S^3$, elements in R_2 can be tuples of any length $n \geq 2$ (but R_2 is finite);
– the input to the reachability problem is a set of goal states $S' \subseteq S$ (instead of a singleton set $\{s'\}$).

This generalisation of AVASS makes it easier to transfer complexity results from AVASS to resource logics, since the transition systems that form the models of resource logics may have more than binary branching, and reachability refers to properties (sets of states) rather singleton states. Here, we will refer to this generalisation as *generalised AVASS*.

Next we briefly elaborate on the concrete reduction of RB±ATL model-checking problem to decision problems on generalised AVASS. Assume that we are designing a state labelling model checking algorithm for RB±ATL, where given a formula ϕ and a model, we label each state with subformulas of ϕ true in that state, in the increasing order of complexity of subformulas. Clearly, there is no problem with doing this for propositional variables and for boolean combinations of earlier encountered formulas, and in fact also for the next state operators. The only difficulty is formulas of the form $\langle\langle A^{\boldsymbol{b}} \rangle\rangle \psi_1 \,\mathcal{U}\, \psi_2$ or $\langle\langle A^{\boldsymbol{b}} \rangle\rangle \Box \psi$. Intuitively, we need to build a different AVASS for every state in the model and every subformula of this form, and then solve a reachability or non-termination problem for them. We describe next how we build this generalised AVASS.

Given an RB-CGS $M = (Agt, Res, S, \Pi, \pi, Act, d, c, \delta)$, a distinguished state s^* (where we want to evaluate the formula) and a coalition $A \subseteq Agt$ (from the main coalition modality in the formula), the corresponding generalised AVASS $G = (S^G, r^G, R_1^G, R_2^G)$ is constructed as follows. The set of states of G is defined as follows:

$$S^G = \{s^*\} \cup \{(s', \alpha) \mid s' \in S, \alpha \in D_A(s')\} \cup \{(\sigma, s'') \mid s'' \in S, \sigma \in D(s'')\}.$$

Obviously, $r^G = Res$. Transitions are defined as follows:

$$R_1^G = \{(s^*, cost(s^*, \alpha), (s^*, \alpha)) \mid \alpha \in D_A(s^*)\} \cup \{((\sigma, s'), cost_A(s', \sigma), (s', \alpha)) \mid (\sigma, s') \in S^G, \alpha \in D_A(s')\}$$
$$R_2^G = \{((s', \alpha), (\sigma^1, s^1), \ldots, (\sigma^k, s^k)) \mid \sigma^i \in D(s'), \alpha = \sigma^i_A, s^i = \delta(s', \sigma^i)\}$$

Note that the size of G is polynomial in M. When evaluating a subformula of the form $\langle\langle A^{\boldsymbol{b}}\rangle\rangle\Box\psi$, the strategy witnessing the truth of the formula has to visit only states satisfying ψ. Since the complexity of ψ is less than the complexity of $\langle\langle A^{\boldsymbol{b}}\rangle\rangle\Box\psi$, we can assume that we know which states in M satisfy ψ. To compute the generalised AVASS where a winning strategy for non-termination exists iff $\langle\langle A^{\boldsymbol{b}}\rangle\rangle\Box\psi$ is true, we remove from S^G all states where the state component of the pair does not satisfy ψ. We denote the resulting generalised AVASS G_ψ. Similarly, to make sure that a strategy to reach a ψ_2 state always goes only through ψ_1 states before reaching ψ_2, we remove from G all states that satisfy neither ψ_1 nor ψ_2. We denote the resulting generalised AVASS G_{ψ_1,ψ_2}.

In [5], Lemmas 2-6 and Theorem 1 demonstrate that $M, s^* \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle\psi_1 \mathcal{U} \psi_2$ if, and only if, there is a winning strategy for Player 1 in a reachability game in the corresponding generalised AVASS G_{ψ_1,ψ_2} with initial credit $\boldsymbol{b}$ and target the set of ψ_2 states, and $M, s^* \models \langle\langle A^{\boldsymbol{b}}\rangle\rangle\Box\psi$ if, and only if, there is a winning strategy for Player 1 in a non-termination game in the the corresponding generalised AVASS G_ψ with initial credit $\boldsymbol{b}$.

3.4 RB±ATL*

RB±ATL* is a more expressive logic than RB±ATL, and was introduced in [5].

As is the case with ATL*, the syntax of RB±ATL* includes state formulas ϕ and path formulas γ. Formulas of RB±ATL* are defined by the following syntax

$$\phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \langle\langle A^{\boldsymbol{b}}\rangle\rangle\gamma$$

$$\gamma ::= \phi \mid \neg\gamma \mid \gamma \vee \gamma \mid \bigcirc\gamma \mid \gamma \mathcal{U} \gamma \mid \Box\gamma \mid$$

where $p \in \Pi$ is a proposition, $A \subseteq Agt$, and $\boldsymbol{b} \in \mathcal{B}$ is a resource bound.

The language of RB±ATL* is interpreted on the same RB-CGS as RB±ATL. The truth definition is identical to that of ATL*, apart from the following clause:

- $M, s \models_s \langle\langle A^{\boldsymbol{b}}\rangle\rangle\gamma$ iff $\exists$ $\boldsymbol{b}$-consistent strategy F_A such that for all $\lambda \in out(s, F_A)$, $M, \lambda \models_p \gamma$;

Definition 14. *The following problem is the model checking problem for RB$\pm$ATL*:*

Input: *A RB-CGS M, a state formula ϕ of RB$\pm$ATL*, and a state $s \in M$.*
Question: *Does it hold that $M, s \models_s \phi$?*

Surprisingly, even without idle actions, which seem to make the difference between decidable and undecidable model-checking for some resource logics (see Section 3.2), the model checking problem for RB$\pm$ATL* is decidable [5] by reduction to parity games on single sided VASS [1]. Moreover, it is decidable in 2EXPTIME, that is, has the same complexity as RB$\pm$ATL.

In [19], several fragments of RB$\pm$ATL and RB$\pm$ATL* of the form RB$\pm$ATL (n, r) and RB$\pm$ATL* (n, r), where the logic is parameterised by the number n of agents and the number r of resource types were studied.[6] In particular, RB$\pm$ATL $(1, 1)$ was shown to be PTIME-complete, and RB$\pm$ATL* $(1, 1)$ PSPACE-complete (see Table 1).

3.5 Other Resource Logics with Decidable Model-Checking

RAL is a very expressive resource logic with undecidable model-checking problem introduced in [23]. In [6], a new syntactic fragment FRAL of RAL with a decidable model-checking problem was identified. FRAL restricts the occurrences of coalitional modalities on the left of Until formulas. On the other hand, it allows nested modalities to refer to resource allocation at the time of evaluation, rather than always considering a fresh resource allocation, as in RB$\pm$ATL. For example, the formula $\langle\!\langle A^{\boldsymbol{b}} \rangle\!\rangle \phi \,\mathcal{U}\, \langle\!\langle A^{\downarrow} \rangle\!\rangle \psi_1 \,\mathcal{U}\, \psi_2$ says that, given resource allocation $\boldsymbol{b}$, coalition A can always reach a state (maintaining ϕ) where, with the remaining resources, it can reach ψ_2 while maintaining ψ_1. In [6] the boundary between decidability and undecidability was also investigated, and the availability of an 'idle' action (i.e., if the semantics requires that in every state each agent has an action that does not produce or consume resources) was shown to be critical: model checking FRAL is decidable in the presence of idle actions, and is not decidable otherwise.

Although model-checking of ATL with perfect recall and uniform strategies is undecidable, if uniformity is replaced with a weaker notion, for example, if it is defined in terms of distributed knowledge, model checking becomes decidable [34]. A similar result hold for RB$\pm$ATSEL, a version of RB$\pm$ATL with syntactic epistemic knowledge and a weaker notion of uniformity [8].

4 Summary and Future Challenges

In Table 1 we summarise the complexity results for the resource logics with a decidable model checking problem discussed in Section 3. In the table, the 'Idle' column indicates whether the semantics for a logic requires that in every state each agent has an action that produces and consumes no resources. New results not appearing in the previous survey [9] are highlighted in bold.

The results for (fragments of) RB$\pm$ATL and RB$\pm$ATL* offer the possibility of significant progress in the verification of resource-bounded multi-agent systems. However many challenges remain for future research. Below we list three of the most important.

[6] Note that RB$\pm$ATL $(n, 1)$ was referred to in [14] as 1-RB$\pm$ATL.

Logic	Resource Production	Idle	Complexity of Model-Checking
RBCL	no	yes	in EXPTIME (PTIME in model) [3]
RB-ATL	no	yes	in EXPTIME (PTIME in model) [4]
PRB-ATL	bounded	yes	EXPTIME-c [32]
RB$\pm$ATL	yes	yes	**2EXPTIME-c** [5]
RB$\pm$ATL $(n, 1)$	yes	yes	in PSPACE [14]
RB$\pm$ATL $(1, 1)$	yes	yes	**PTIME-c** [19]
FRAL	yes	yes	?
RB$\pm$ATSEL	yes	yes	?
RB$\pm$ATL*	yes	no	**2EXPTIME-c** [5]
RB$\pm$ATL* $(n, 1)$	yes	no	**EXPSPACE-c** [5]
RB$\pm$ATL* $(1, 1)$	yes	yes	**PSPACE-c** [5, 19]

Table 1: Resource logics with decidable model-checking problem

Understanding the Sources of Undecidability Developing a better understanding of the sources of decidability and undecidability (beyond boundedness) will be critical to future progress. As observed in [23], subtle differences in truth conditions for resource logics result in the difference between decidability and undecidability of the model checking problem. Some work in this direction is reported in [6, 7, 5].

Logics with Lower Complexity It is useful to discover sources of undecidability and how to construct expressive logics for which the model-checking problem is decidable. However, it is even more important to be able to develop logics, or fragments of existing logics such as RB$\pm$ATL, that are sufficiently expressive for practical problems, and where the model-checking problem has tractable complexity. Only then will we be able to implement practical model-checking tools for systems of resource-bounded agents.

Practical Tools Although model checking algorithms have been proposed for several of the logics surveyed, work on implementation is only beginning. We aim to develop practical model-checking tools for verifying resource-bounded MAS by extending the MCMAS model checker [41] to allow the modelling of multi-agent systems in which agents can both consume and produce resources. Work on symbolic encoding of RB-ATL model-checking is reported in [15] and work on symbolic encoding of RB$\pm$ATL model-checking is reported in [13].

Addressing these challenges will allow practical model-checking of resource logics and significant advances in multi-agent system verification.

Acknowledgements The authors thank Stéphane Demri for helpful discussions.

References

1. Abdulla, P., Mayr, R., Sangnier, A., Sproston, J.: Solving Parity Games on Integer Vectors. In: CONCUR'13. Lecture Notes in Computer Science, vol. 8052, pp. 106–120. Springer (2013)
2. Abdulla, P.A., Bouajjani, A., d'Orso, J.: Deciding monotonic games. In: Baaz, M., Makowsky, J.A. (eds.) Computer Science Logic, 17th International Workshop, CSL 2003, 12th Annual Conference of the EACSL, and 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2803, pp. 1–14. Springer (2003)
3. Alechina, N., Logan, B., Nguyen, H.N., Rakib, A.: A logic for coalitions with bounded resources. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009). vol. 2, pp. 659–664. IJCAI/AAAI, AAAI Press (2009)
4. Alechina, N., Logan, B., Nguyen, H.N., Rakib, A.: Resource-bounded alternating-time temporal logic. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010). pp. 481–488. IFAAMAS (2010)
5. Alechina, N., Bulling, N., Demri, S., Logan, B.: On the complexity of resource-bounded logics. Theoretical Computer Science 750, 69–100 (2018), `https://doi.org/10.1016/j.tcs.2018.01.019`
6. Alechina, N., Bulling, N., Logan, B., Nguyen, H.N.: On the boundary of (un)decidability: Decidable model-checking for a fragment of resource agent logic. In: Yang, Q. (ed.) Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015). IJCAI, AAAI Press, Buenos Aires, Argentina (July 2015)
7. Alechina, N., Bulling, N., Logan, B., Nguyen, H.N.: The virtues of idleness: A decidable fragment of resource agent logic. Artificial Intelligence 245, 56–85 (2017), `http://dx.doi.org/10.1016/j.artint.2016.12.005`
8. Alechina, N., Dastani, M., Logan, B.: Verifying existence of resource-bounded coalition uniform strategies. In: Rossi, F. (ed.) IJCAI 2016, Proceedings of the 25th International Joint Conference on Artificial Intelligence. IJCAI/AAAI (2016)
9. Alechina, N., Logan, B.: Verifying systems of resource-bounded agents. In: Beckmann, A., Bienvenu, L., Jonoska, N. (eds.) Pursuit of the Universal: 12th Conference on Computability in Europe, CiE 2016, Paris, France, June 27 - July 1, 2016, Proceedings. pp. 3–12. Springer (2016)
10. Alechina, N., Logan, B.: Resource logics with a diminishing resource (extended abstract). In: Dastani, M., Sukthankar, G., Andre, E., Koenig, S. (eds.) Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018). pp. 1847–1849. IFAAMAS (2018)
11. Alechina, N., Logan, B., Nga, N.H., Rakib, A.: Logic for coalitions with bounded resources. Journal of Logic and Computation 21(6), 907–937 (December 2011)
12. Alechina, N., Logan, B., Nguyen, H.N., Raimondi, F.: Decidable model-checking for a resource logic with production of resources. In: Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-2014). pp. 9–14. ECCAI, IOS Press, Prague, Czech Republic (August 2014)
13. Alechina, N., Logan, B., Nguyen, H.N., Raimondi, F.: Symbolic model-checking for one-resource RB+-ATL. In: Yang, Q. (ed.) Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015). IJCAI, AAAI Press, Buenos Aires, Argentina (July 2015)
14. Alechina, N., Logan, B., Nguyen, H.N., Raimondi, F.: Model-checking for resource-bounded ATL with production and consumption of resources. Journal of Computer and System Sciences 88, 126–144 (September 2017)

15. Alechina, N., Logan, B., Nguyen, H.N., Raimondi, F., Mostarda, L.: Symbolic model-checking for resource-bounded ATL. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015. pp. 1809–1810. ACM (2015)
16. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM 49(5), 672–713 (2002)
17. Alur, R., Henzinger, T.A., Mang, F.Y.C., Qadeer, S., Rajamani, S.K., Tasiran, S.: MOCHA: Modularity in model checking. In: Computer Aided Verification. pp. 521–525 (1998)
18. Belardinelli, F.: Verification of non-uniform and unbounded artifact-centric systems: decidability through abstraction. In: Bazzan, A.L.C., Huhns, M.N., Lomuscio, A., Scerri, P. (eds.) International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14. pp. 717–724. IFAAMAS/ACM (2014)
19. Belardinelli, F., Demri, S.: Resource-bounded ATL: the quest for tractable fragments. In: Elkind, E., Veloso, M., Agmon, N., Taylor, M.E. (eds.) Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19. pp. 206–214. International Foundation for Autonomous Agents and Multiagent Systems (2019)
20. Bordini, R.H., Fisher, M., Visser, W., Wooldridge, M.: Model checking rational agents. IEEE Intelligent Systems 19(5), 46–52 (2004)
21. Bordini, R.H., Fisher, M., Visser, W., Wooldridge, M.: Verifying multi-agent programs by model checking. Journal of Autonomous Agents and Multi-Agent Systems 12(2), 239–256 (March 2006), `http://dro.dur.ac.uk/622/`
22. Brázdil, T., Jančar, P., Kucera, A.: Reachability games on extended vector addition systems with states. In: ICALP'10. Lecture Notes in Computer Science, vol. 6199, pp. 478–489. Springer (2010)
23. Bulling, N., Farwer, B.: On the (un-)decidability of model checking resource-bounded agents. In: Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010). Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 567–572. IOS Press (2010)
24. Bulling, N., Farwer, B.: On the (un-)decidability of model checking resource-bounded agents. Tech. Rep. IfI-10-05, Clausthal University of Technology (2010)
25. Bulling, N., Farwer, B.: Expressing properties of resource-bounded systems: The logics RBTL and RBTL*. In: Post-Proceedings of CLIMA '09. pp. 22–45. LNCS 6214 (2010)
26. Bulling, N., Goranko, V.: How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In: Mogavero, F., Murano, A., Vardi, M.Y. (eds.) Proceedings 1st International Workshop on Strategic Reasoning, SR 2013. EPTCS, vol. 112, pp. 33–41 (2013)
27. Bulling, N., Nguyen, H.N.: Model checking resource bounded systems with shared resources via alternating büchi pushdown systems. In: Chen, Q., Torroni, P., Villata, S., Hsu, J.Y., Omicini, A. (eds.) PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference. Lecture Notes in Computer Science, vol. 9387, pp. 640–649. Springer (2015), `http://dx.doi.org/10.1007/978-3-319-25524-8`
28. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.: Generalized mean-payoff and energy games. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India. LIPIcs, vol. 8, pp. 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
29. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems 8(2), 244–263 (1986)
30. Courtois, J., Schmitz, S.: Alternating vector addition systems with states. In: MFCS'14. Lecture Notes in Computer Science, vol. 8634, pp. 220–231. Springer (2014)

31. Della Monica, D., Napoli, M., Parente, M.: On a logic for coalitional games with priced-resource agents. Electr. Notes Theor. Comput. Sci. 278, 215–228 (2011)
32. Della Monica, D., Napoli, M., Parente, M.: Model checking coalitional games in shortage resource scenarios. In: Proceedings of the 4th International Symposium on Games, Automata, Logics and Formal Verification (GandALF 2013. EPTCS, vol. 119, pp. 240–255 (2013)
33. Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.: Model checking agent programming languages. Automated Software Engineering 19(1), 5–63 (2012)
34. Dima, C., Tiplea, F.L.: Model-checking ATL under imperfect information and perfect recall semantics is undecidable. CoRR abs/1102.4225 (2011), `http://arxiv.org/abs/1102.4225`
35. Fisher, M., Dennis, L.A., Webster, M.P.: Verifying autonomous systems. Commun. ACM 56(9), 84–93 (2013)
36. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
37. Jurdzinski, M., Lazic, R., Schmitz, S.: Fixed-dimensional energy games are in pseudo-polynomial time. In: Proceedings of ICALP 2015. pp. 260–272 (2015)
38. Kanovich, M.I.: Petri nets, horn programs, linear logic and vector games. Ann. Pure Appl. Logic 75(1-2), 107–135 (1995)
39. Kwiatkowska, M.Z., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proceedings of 23rd International Conference on Computer Aided Verification (CAV 2011). Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011)
40. Lincoln, P., Mitchell, J.C., Scedrov, A., Shankar, N.: Decision problems for propositional linear logic. Ann. Pure Appl. Logic 56(1-3), 239–311 (1992)
41. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multi-agent systems. In: Bouajjani, A., Maler, O. (eds.) Proceedings of the 21st International Conference on Computer Aided Verification (CAV 2009. Lecture Notes in Computer Science, vol. 5643, pp. 682–688. Springer (2009)
42. Nguyen, H.N., Rakib, A.: A probabilistic logic for resource-bounded multi-agent systems. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 521–527. ijcai.org (2019)
43. Raskin, J., Samuelides, M., Begin, L.V.: Games for counting abstractions. Electr. Notes Theor. Comput. Sci. 128(6), 69–85 (2005)
44. Shapiro, S., Lespérance, Y., Levesque, H.J.: The cognitive agents specification language and verification environment for multiagent systems. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02). pp. 19–26. ACM Press, New York, NY, USA (2002)