ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-567

On the (Un-)Decidability of Model Checking Resource-Bounded Agents

Nils Bulling[1] and **Berndt Farwer**[2]

Abstract. The verification and modelling of multi-agent systems is an important topic that has attracted much attention in recent years. Resources, however, have only recently been studied as simple extensions to well-known logics. Trying to find a set of useful features while retaining essential properties for practical use, we explore the question: *Where are the limits of what can be verified about resource-bounded agents?* We try to answer this question by considering several natural logic-based settings that may arise and prove that verification is usually *undecidable* apart from bounded or otherwise restrictive settings. Most interestingly, we identify various factors that influence the (un-)decidability and provide grounds for future research on more promising constraints leading to decidable fragments.

1 Introduction

Verification of multi-agent systems, in particular the *model-checking problem* (i.e. whether a given property holds in a given model), has attracted much attention in recent years [9, 10, 4, 13, 15, 12]. Most of these results focus on well-established logics like the computation tree logics or alternating time temporal logics [10, 4]. Just recently these logics have been extended to verify various aspects of *rational* agents [8, 7]. However, the basic idea of rational agents being autonomous entities perceiving changes in their environment and acting according to a set of rules or plans in the pursuit of goals does not take into account resources. But many actions that an agent would execute in order to achieve a goal can – in real life – only be carried out in the presence of certain resources. Without sufficient resources some actions are not available, leading to plan failure. The analysis and verification of agent systems with resources of this kind is still in its infancy; the only work we are aware of in this direction is [5, 2, 1].

In this paper we investigate the boundaries of what can and cannot be verified about resource-bounded agents. It turns out that the handling of resources is harder than it may seem at first sight: we prove that in many settings the model-checking problem is undecidable.

The paper is structured as follows. In Section 2, we introduce the general language and semantics, as well as some restricted variants. Section 3 presents one of the main contributions of this paper; we discuss the model-checking problems for various settings of our logic. Finally, we conclude the paper with a discussion of related and future work.

[1] TU Clausthal , Germany, email:bulling@in.tu-clausthal.de
[2] Durham University, UK, email:berndt.farwer@durham.ac.uk

2 Resource-Bounded Agent Logic

This section introduces the logic RAL^* (*Resource-Bounded Agent Logic*), *Resource-Bounded Models* (**RBM**s), and restricted variants of the logic. In the following we assume that $\mathbb{A}\text{gt} = \{1, \dots, k\}$ is a finite set of *agents*, Q is a finite set of *states*, $\mathcal{R}es = \{R_1, \dots, R_u\}$ is a finite set of *resource types* or just *resources*, and $\mathcal{P}rops = \{\mathsf{p}, \mathsf{q}, \dots\}$ is a set of propositions. We often use "$a, b, \dots$" and "$A, B, \dots$" to refer to agents (i.e. $a, b \in \mathbb{A}\text{gt}$) and groups of agents (i.e. $A, B \subseteq \mathbb{A}\text{gt}$), respectively.

We use an *endowment* function $\eta : \mathbb{A}\text{gt} \times \mathcal{R}es \to \mathbb{N}_0^\infty$ to model the amount of resources an agent is equipped with[3]: $\eta(a, r)$ is the amount of resource r agent a possesses. The set of all endowments is denoted by En. We also write η_a for $\eta(a)$. The quantity "∞" is used to equip an agent with an infinite amount of resources. This allows us to ignore specific resource types for that agent. We define the endowment η^∞ as the constant function that maps every resource for every agent to ∞. Finally, we use a *resource-quantity mapping* (**rqm**) $\rho : \mathcal{R}es \to \mathbb{Z}^\infty$ to model the currently available resources (in the system); that is, $\rho(r)$ denotes to availability or lack of resource r.

The Language. From the syntactic perspective the logic RAL^*, introduced in the following, is not much different from the *alternating-time temporal logic* ATL^* [4]. Cooperation modalities come in two versions: $\langle\langle A\rangle\rangle_B$ and $\langle\langle A\rangle\rangle_B^\eta$ where $A, B \subseteq \mathbb{A}\text{gt}$. For both modalities it is assumed that agents in $A \cup B$ require resources. The reading of $\langle\langle A\rangle\rangle_B^\eta\gamma$ is that *agents A have a strategy compatible with the endowment η to enforce γ*. The operator $\langle\langle A\rangle\rangle_B\gamma$ reads similarly but the strategy must be compatible with the resources currently (implicitly) available to the agents. That is, the former operator equips the agents with a *fresh* amount of resources.

Definition 1 (Language $\mathcal{L}_{\mathbf{RAL}^*}$) *The language $\mathcal{L}_{RAL^*}$ is defined as follows*[4]*: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle_B\gamma \mid \langle\langle A\rangle\rangle_B^\eta\gamma$ where $\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \varphi\,\mathcal{U}\,\varphi \mid \bigcirc\varphi$, $A, B \subseteq \mathbb{A}\text{gt}$, $\mathsf{p} \in \mathcal{P}rops$, and*[5] *$\eta \in \mathsf{En}$. Formula φ (resp. γ) is called* state *formula (resp.* path *formula). Moreover, we use $\langle\langle A\rangle\rangle^\eta$ (resp. $\langle\langle A\rangle\rangle$) as an abbreviation for $\langle\langle A\rangle\rangle_A^\eta$ (resp. $\langle\langle A\rangle\rangle_A$).*

The temporal operators $\bigcirc$ and $\mathcal{U}$ have the standard meaning *'in the next moment'* and *'until'*, respectively. As usual, one defines $\Diamond\,\gamma \equiv \top\,\mathcal{U}\,\gamma$ (*eventually*) and $\Box\,\gamma = \neg\Diamond\,\neg\gamma$ (*always from now on*).

[3] $\mathbb{N}_0^\infty$ (resp. $\mathbb{Z}^\infty$) is defined as $\mathbb{N}_0 \cup \{\infty\}$ (resp. $\mathbb{Z} \cup \{\infty\}$).
[4] Due to the lack of space, we also use semantic symbols in the object language.
[5] As we are mainly interested in decidability results in this paper the concrete representation of η is irrelevant.

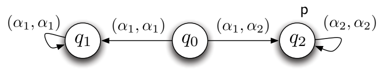

Figure 1. A simple **RBM** $\mathfrak{M}$ with $\mathbb{A}\mathrm{gt} = \{1, 2\}$, $d_1(q_0) = d_1(q_1) = d_2(q_1) = \{\alpha_1\}$, $d_1(q_2) = d_2(q_2) = \{\alpha_2\}$, $d_2(q_0) = \{\alpha_1, \alpha_2\}$ and one resource R. Actions α_1 costs one unit of R and action α_2 is cost-free; i.e. $t(\alpha_1, R) = -1$ and $t(\alpha_2, R) = 0$.

The Semantics. As models for our logic we take concurrent game structures (**CGS**) form [4] and extend them by resources and a mapping t indicating how many resources each action requires or produces when executed.

Definition 2 (RBM) *A* resource-bounded model *(**RBM**) is given by $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \mathcal{P}rops, \pi, Act, d, o, \mathcal{R}es, t)$ where $\pi : Q \to \mathcal{P}(\mathcal{P}rops)$ is a* valuation of propositions*; Act is a finite set of* actions*; and function $d : \mathbb{A}\mathrm{gt} \times Q \to \mathcal{P}(Act)$ indicates the actions available to agent $a \in \mathbb{A}\mathrm{gt}$ in state $q \in Q$. We write $d_a(q)$ instead of $d(a, q)$, and use $d(q)$ to denote the set $d_1(q) \times \cdots \times d_k(q)$ of* action profiles *in state q. o is a serial* transition function *which maps each state $q \in Q$ and action profile $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_k \rangle \in d(q)$ (specifying a move for* each *agent) to another state $q' = o(q, \vec{\alpha})$. Finally, the function $t : Act \times \mathcal{R}es \to \mathbb{Z}$ models the resources consumed and produced by an action. We define $\mathsf{prod}(\alpha, r) := \max\{0, t(\alpha, r)\}$ (resp. $\mathsf{cons}(\alpha, r) := -\min\{0, t(\alpha, r)\}$) as the amount of resource r produced (resp. consumed) by action α..*

For $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_k \rangle$, we use $\vec{\alpha}|_A$ to denote the sub-tuple consisting of the actions of agents $A \subseteq \mathbb{A}\mathrm{gt}$ and we use $X_{\mathfrak{M}}$ to refer to an element X contained in $\mathfrak{M}$.

Example 1 *Figure 1 shows a simple **RBM**.*

Note that the tuple $(\mathbb{A}\mathrm{gt}, Q, \mathcal{P}rops, \pi, Act, d, o)$ is simply a concurrent game structure as introduced in [4]. We define $Q^{\leq\omega} := Q^\omega \cup Q^+$ (i.e. all infinite and finite sequences over Q). A *path* $\lambda \in Q^{\leq\omega}$ is a finite or infinite sequence of states such that there is a transition between two adjacent states. Intuitively, not all paths are possible given limited resources. We define a *resource extended path* λ as a finite or infinite sequence over $Q \times \mathsf{En}$ such that the restriction of λ to states (the first component), denoted by $\lambda|_Q$, is a path in the underlying model. Similarly, we use $\lambda|_{\mathcal{R}es}$ to refer to the projection of λ to the second component of each element in the sequence.

We also use the following notations introduced for paths. The *length* of λ (where λ is a path or resource extended path), denoted $l(\lambda)$, is the number of states in the sequence; if $\lambda \in Q^\omega$ then $l(\lambda) = \infty$. For $i \in \mathbb{N}_0$, we define $\lambda[i]$ to be the $(i+1)$-th sate on λ or the last one if $i \geq l(\lambda)$. Moreover, $\lambda[i, \infty]$ refers to the infinite subpath $\lambda[i]\lambda[i{+}1] \ldots$ of λ if $l(\lambda) = \infty$; or to the finite subpath $\lambda[i]\lambda[i+1] \ldots \lambda[l(\lambda) - 1]$ if $l(\lambda) < \infty$. The set of all paths in $\mathfrak{M}$ starting in a state q is defined by $\Lambda_{\mathfrak{M}}(q)$.

Ultimately, we are interested in the ability of groups of agents. We are interested in the existence of a *winning strategy* for a group of agents. A *strategy* is a function that fixes the behaviour of an agent; that is, it determines an action for each 'situation' where we will consider two types of situations. Once, agents can base their decision on the current state only and once, on the whole previous history.

Definition 3 (R/r-strategy) *A* perfect-recall strategy for agent a *(or* R-strategy*) is a function $s_a : Q^+ \to Act$ such that $s_a(q_1 \ldots q_n) \in d_a(q_n)$. A strategy s_a is called* memoryless *(or* r-strategy*) if $s_A(hq) = s_A(h'q)$ for all $h, h' \in Q^*$ and $q \in Q$ (such strategies can be defined as functions $Q \to Act$).*

The condition "$s_a(q_1 \ldots q_n) \in d_a(q_n)$" ensures that the prescribed action is executable by the agent.

Actions require or produce certain amounts of resources (modelled by t) that have to be present for an action to be executed. Agents in a group A can cooperate and *share* their resources, as can the opponents $\mathbb{A}\mathrm{gt}\backslash A$. In the following, we formalise such 'shares' sh with respect to an available endowment η for some **rqm** ρ.

Definition 4 (**(A, η)-share for** ρ) *Let η be an endowment and let ρ be an **rqm**. An (A, η)*-share for *ρ is a function $\mathsf{sh} : A \times \mathcal{R}es \to \mathbb{N}_0$ such that:*

1. *$\forall r \in \mathcal{R}es : \rho(r) > 0 \Rightarrow \sum_{a \in A} \mathsf{sh}(a, r) = \rho(r)$ (the share equals the demand); and*
2. *$\forall a \in A, r \in \mathcal{R}es : \eta_a(r) \geq \mathsf{sh}(a, r)$ (each agent's share must be available).*

A strategy s_A restricts the possible paths in an **RBM**; moreover, considering resource-extended paths, only those in which agents have sufficiently resources available in each state are feasible. We use the resource component to keep track of the available resources.

We define which extended paths λ are possible under a given endowment η and strategy s_A assuming agents $A \cup B$ require resources.

Definition 5 (**(η, s_A, B)-path,** $out(q, \eta, s_A, B)$) *An (η, s_A, B)-path is a* maximal *resource-extended path $\lambda \in (Q \times \mathsf{En})^{\leq\omega}$ such that for all $i = 0, \ldots, l(\lambda) - 1$ with $\lambda[i] := (q_i, \eta^i)$ there is an action profile $\vec{\alpha} \in d(\lambda|_Q[i])$ such that*

1. *$\lambda|_{\mathcal{R}es}[0] \leq \eta$ (initially at most η resources are available)*
2. *$s_A(\lambda|_Q[0, i]) = \vec{\alpha}|_A$ (A's actions in $\vec{\alpha}$ are the ones prescribed by strategy s_A),*
3. *$\lambda|_Q[i+1] = o(\lambda|_Q[i], \vec{\alpha})$ (transitions are taken according to the action profile $\vec{\alpha}$),*
4. *$\forall\ a \in A \forall r \in \mathcal{R}es : (\eta_a^{i+1}(r) = \eta_a^i(r) + \mathsf{prod}(\vec{\alpha}|_a, r) - \mathsf{sh}(a, r))$ where $\mathsf{sh} : A \times \mathcal{R}es \to \mathbb{N}_0$ is an (A, η)-share for $r \mapsto \sum_{a \in A} \mathsf{cons}(\vec{\alpha}|_a, r)$ (A's resources change according to some appropriate share),*
5. *$\forall b \in B\backslash A\ \forall r \in \mathcal{R}es : (\eta_b^{i+1}(r) = \eta_b^i(r) + \mathsf{prod}(\vec{\alpha}|_b, r) - \mathsf{sh}(b, r))$ where $\mathsf{sh} : B\backslash A \times \mathcal{R}es \to \mathbb{N}_0$ is an $(B\backslash A, \eta)$-share for $r \mapsto \sum_{b \in B\backslash A} \mathsf{cons}(\vec{\alpha}|_b, r)$) ($B\backslash A$'s resources change according to some appropriate share),*
6. *$\forall a \in \mathbb{A}\mathrm{gt}\backslash(A \cup B)\ \forall r \in \mathcal{R}es : (\eta_a^{i+1}(r) = \eta_a^i(r))$ (available resources remain unchanged for all agents not in $A \cup B$),*
7. *$\forall a \in \mathbb{A}\mathrm{gt} : (\lambda|_{\mathcal{R}es}[i]_a \geq 0 \Rightarrow \lambda|_{\mathcal{R}es}[i+1]_a \geq 0)$ and $(\lambda|_{\mathcal{R}es}[i]_a < 0 \Rightarrow \lambda|_{\mathcal{R}es}[i+1]_a \geq \lambda|_{\mathcal{R}es}[i+1]_a$ (for each step the required resources are available).*

The η-outcome *of a strategy s_A against B in q, $out(q, \eta, s_A, B)$, is defined as the set of all (η, s_A, B)-paths starting in q.*

Remark 1 *(1) We require that a path is maximal, i.e., if a given path can be extended (this is the case if sufficient resources are available) then it must be extended. (2) After an action has been executed the production of resources is added to the endowment of the* action-executing agent*. (3) There are several (η, s_A, B)-paths due to the choices of the opponents* and *due to different shares in items 4 and 5.*

Finally, we define four semantics for $\mathcal{L}_{\mathrm{RAL}^*}$ over triples of an **RBM** together with a state and a given endowment for the agents.

Definition 6 (**$\models_R$, $\models_r$, $\models_R^\infty$, $\models_r^\infty$, RAL_R^*, RAL_r^***) *Consider an **RBM** $\mathfrak{M}$, a state $q \in Q_{\mathfrak{M}}$, and an endowment η. The R*-semantics *is given by the satisfaction relation $\models_R$ defined as follows.*

$\mathfrak{M}, q, \eta \models_R \mathsf{p}$ *iff* $\mathsf{p} \in \pi(q)$
$\mathfrak{M}, q, \eta \models_R \neg\varphi$ *iff* $\mathfrak{M}, q, \eta \not\models_R \varphi$
$\mathfrak{M}, q, \eta \models_R \varphi \wedge \psi$ *iff* $\mathfrak{M}, q, \eta \models_R \varphi$ *and* $\mathfrak{M}, q, \eta \models_R \psi$
$\mathfrak{M}, q, \eta \models_R \langle\langle A \rangle\rangle_C \gamma$ *iff there is an R-strategy* s_A *for* A *such that* $\mathfrak{M}, \lambda, \eta \models_R \gamma$ *for all* $\lambda \in out(q, \eta, s_A, C)$
$\mathfrak{M}, q, \eta \models_R \langle\langle A \rangle\rangle^{\zeta}_C \gamma$ *iff there is an R-strategy* s_A *for* A *such that* $\mathfrak{M}, \lambda, \zeta \models_R \gamma$ *for all* $\lambda \in out(q, \zeta, s_A, C)$
$\mathfrak{M}, \lambda, \eta \models_R \varphi$ *iff* $\mathfrak{M}, \lambda[0], \eta \models_R \varphi$

and for path formulae

$\mathfrak{M}, \lambda, \eta \models_R \neg\gamma$ *iff not* $\mathfrak{M}, \lambda, \eta \models_R \gamma$
$\mathfrak{M}, \lambda, \eta \models_R \gamma \wedge \chi$ *iff* $\mathfrak{M}, \lambda, \eta \models_R \gamma$ *and* $\mathfrak{M}, \lambda, \eta \models_R \chi$
$\mathfrak{M}, \lambda, \eta \models_R \bigcirc\gamma$ *iff* $\mathfrak{M}, \lambda[1, \infty], \lambda|_{\mathcal{R}es}[1] \models_R \gamma$ *and* $l(\lambda) > 1$
$\mathfrak{M}, \lambda, \eta \models_R \gamma \mathcal{U} \chi$ *iff there is* $i < l(\lambda)$ *such that* $\mathfrak{M}, \lambda[i, \infty], \lambda|_{\mathcal{R}es}[i] \models_R \chi$ *and for all* j *with* $0 \leq j < i$ *we have* $\mathfrak{M}, \lambda[j, \infty], \lambda|_{\mathcal{R}es}[j] \models_R \gamma$

The r-semantics *(memoryless semantics)* $\models_r$ *is defined similarly to the R-semantics but* r-strategies *are used instead of R-strategies. Moreover, we introduce a variant that focuses on infinite paths. Therefore, in the semantic clauses of the cooperation modalities, we replace "*$\lambda \in out(q, \eta, s_A, C)$*" with "infinite* $\lambda \in out(q, \zeta, s_A, C)$*". The resulting semantic relations are denoted* $\models^{\infty}_R$ *and* $\models^{\infty}_r$*.*

The logic $\textbf{RAL}^*_R$ *(resp.* $\textbf{RAL}^*_r$*) is defined as the language* $\mathcal{L}_{RAL^*}$ *together with R-semantics* $\models_R$ *(resp. r-semantics* $\models_r$*).*

The 'infinite semantics' is needed for some extended expressivity and complexity results. The language $\mathcal{L}_{\mathrm{RAL}^*}$, however, is sufficiently expressive to describe infinite paths by "$\Box \bigcirc \top \rightarrow \ldots$", so we can state as a fact that the semantics focusing on infinite paths can be simulated by $\mathcal{L}_{\mathrm{RAL}^*}$ (for a proof, see [6]).

Proposition 1 *Logic* $(\mathcal{L}_{RAL^*}, \models_x)$ *subsumes* $(\mathcal{L}_{RAL^*}, \models^{\infty}_x)$ *for* $x \in \{r, R\}$*. This also holds for the* proponent restrictive *(pr) and* resource flat *(rf) variants of Definition 8.*

Example 2 *Recall the* **RBM** *from Example 1 and consider the following endowment* η*:* $\eta(1)(R) = 2$ *and* $\eta(2)(R) = \infty$*. Then, we have* $\mathfrak{M}, q_0, \eta \not\models_r \langle\langle 1 \rangle\rangle \Diamond \mathsf{p}$ *and* $\mathfrak{M}, q_0, \eta \models_r \langle\langle 2 \rangle\rangle \Diamond \mathsf{p}$*; there are two paths* λ *and* λ' *in the outcome:* $\lambda|_Q = q_0(q_2)^{\omega}$ *and* $\lambda'|_Q = q_0q_1q_1$*. But note, that we have* $\mathfrak{M}, q_0, \eta \models^{\infty}_r \langle\langle 1 \rangle\rangle \Diamond \mathsf{p}$ *as the finite path* λ' *is disregarded.*

Syntactically Restricted Variants. Following [10, 4], we define (temporal) restrictions of $\mathcal{L}_{\mathrm{RAL}^*}$.

Definition 7 (Languages $\mathcal{L}_{\mathbf{RAL}^+}$ and $\mathcal{L}_{\mathbf{RAL}}$) *The language* $\mathcal{L}_{RAL^+}$ *restricts* $\mathcal{L}_{RAL^*}$ *in such a way that path formulae are given by* $\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \varphi \mathcal{U} \varphi \mid \bigcirc\varphi$*.*
The language $\mathcal{L}_{RAL}$ *is given by*

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A \rangle\rangle_B \bigcirc \varphi \mid \langle\langle A \rangle\rangle_B \Box \varphi \mid \langle\langle A \rangle\rangle_B \varphi \mathcal{U} \varphi \mid \langle\langle A \rangle\rangle_B \varphi \mathcal{R} \varphi \mid \langle\langle A \rangle\rangle^{\eta}_B \bigcirc \varphi \mid \langle\langle A \rangle\rangle^{\eta}_B \Box \varphi \mid \langle\langle A \rangle\rangle^{\eta}_B \varphi \mathcal{U} \varphi \mid \langle\langle A \rangle\rangle^{\eta}_B \varphi \mathcal{R} \varphi$$

For the semantic interpretation we consider the 'release' operator as the following macro: $\varphi \mathcal{R} \psi \equiv \neg((\neg\psi) \mathcal{U} (\neg\varphi))$. Differently, from [10, 4] in the case of $\mathcal{L}_{CTL}$ and $\mathcal{L}_{\mathrm{ATL}}$we do also allow the 'release' operator $\mathcal{R}$. Note that $\mathcal{L}_{\mathrm{RAL}}$ with the release operator is strictly more expressive than it would be without it [14].

Next, we define variants of all languages that restrict the use of resources. Operators $\langle\langle A \rangle\rangle_B$ assume that the proponents A *and* opponents $B \backslash A$ act under limited resources whereas $\langle\langle A \rangle\rangle$ only restricts the choices of the proponents A. In Section 3 we show that this influences the model checking complexity.

Another aspect of complexity is reflected by the two cooperation modalities $\langle\langle A \rangle\rangle_C$ and $\langle\langle A \rangle\rangle^{\eta}_C$. The former operator is computationally harder to handle than the latter as one has to keep track of resources. Note, that the expressiveness of the logic justifies operators of the first kind. For example, consider the formula $\langle\langle A \rangle\rangle \Diamond (\mathsf{p} \wedge \langle\langle B \rangle\rangle \gamma)$: agents A have to reach a state in which p holds and in which B can ensure γ with the then *remaining resources* for agents $A \cap B$.

Both restrictions have interesting effects on the model checking complexity and the number of agents needed to show undecidability.

Definition 8 (Proponent restrictiveness; resource flatness) *Let* $\mathcal{L}$ *be any of the languages introduced so far.*

(a) The language pr-$\mathcal{L}$*,* proponent-restricted $\mathcal{L}$ *, is the sublanguage of* $\mathcal{L}$ *allowing only operators* $\langle\langle A \rangle\rangle$ *and* $\langle\langle A \rangle\rangle^{\eta}$*.*
(b) The language rf-$\mathcal{L}$*,* resource-flat $\mathcal{L}$ *, is the sublanguage obtained from* $\mathcal{L}$ *if only cooperation modalities* $\langle\langle A \rangle\rangle^{\eta}_B$ *are allowed (and not* $\langle\langle A \rangle\rangle_B$*).*

Analogously to Definition 6, we define the logics $\textbf{RAL}_R$*,* $\textbf{RAL}_r$*,* $\textbf{RAL}^+_r$*, and* $\textbf{RAL}^+_R$ *and their proponent-restricted and/or resource-flat variants.*

Restricted RBMs. In Section 3 we show that the model-checking problem is often undecidable over general **RBM**s. Exceptions are two bounded settings presented in the following.

For $k \in \mathbb{N}$, an **RBM** $\mathfrak{M}$ is said to be *k-bounded for endowment* η if for every element (q, ζ) on any (η, s_A, B)-path for any strategy s_A and $B \subseteq \mathbb{A}\mathrm{gt}$ either $\zeta(a)(r) \leq k$ or $\zeta(a)(r) = \infty$ holds for all resources $r \in \mathcal{R}es_{\mathfrak{M}}$ and agents $a \in \mathbb{A}\mathrm{gt}$. An **RBM** is called *bounded for* η if it is k-bounded for η for some $k \in \mathbb{N}$.

At a first glance such models may seem quite artificial but in fact there are several natural settings resulting in bounded models. We call a model *production-free* if actions can only consume and not produce resources. Clearly, every production-free model is bounded.

There is another way to *enforce* a bounded setting. The definition above is purely structural and obviously not every **RBM** is bounded. However, often agents themselves have limited capabilities such that it does not necessarily make sense to allow them to carry arbitrary amounts of resources. Depending on the resource type only a limited number of units may be permitted in any endowment. In this setting one *imposes* the requirement of boundedness to the semantics and simply discards any resources that exceed a given bound. The latter is a *semantic restriction* and has to be inserted into the definition of paths.

We define a *k-bounded* (η, s_A, B)*-path* as in Definition 5 but we set set $\lambda|_{\mathcal{R}es}[0] = \eta^1_a(r) := \min\{k, \eta^i_a(r)\}$ and replace conditions 4 and 5 by the following: $\eta^{i+1}_a(r) = \min\{k, \eta^i_a(r) + \mathsf{prod}(\vec{\alpha}|_a, r) - \mathsf{sh}(a, r))\}$.

The *k-bounded* η*-outcome* of a strategy s_A in q with respect to B, $out_k(q, \eta, s_A, B)$, is defined as the set of all k-bounded (η, s_A, B)-paths starting in q.

Finally, we define the *k-bounded* R-semantics $\models^k_R$ (resp. r-semantics $\models^k_r$) as in Definition 6 but replace the outcome by the k-outcome.

3 Verification: (Un-)Decidability

In this section we turn to the model-checking problem and consider how the variously restricted settings influence its complexitiy. Full proofs can be found in [6].

Decidability Results. For both bounded settings introduced in Section 2 we have that along each resource extended path there are only finitely many reachable states from $Q \times \mathsf{En}$. Hence, given an endowment, we can 'unravel' a given **RBM** and apply 'normal' **ATL*** model checking [4] which is proven to be decidable. Note, however, that the unraveling may yield finite paths (i.e. states with no successor) requiring a straightforward extension of existing algorithms.

Theorem 1 *Model checking **RAL**$^*_R$ (and all other variants discussed here) is decidable over the class of bounded **RBM**s and the k-bounded semantics.*

Undecidability Results. In this section, we consider all settings apart from the bounded ones. It is well known that the model-checking problems for **ATL**$_R$, **ATL**$^*_r$, and **ATL**$^*_R$ are $\mathbf{P}$-complete, $\mathbf{PSPACE}$, and $\mathbf{2EXPTIME}$-complete, respectively [4]. Model checking **RTL** is shown decidable in [5], and the same holds for **RBCL** [2]. Here, we show that the latter two cases form an exception; the general resource-bounded settings turn out to be undecidable due to the possibility of *producing* resources.

Two-Counter Automata. The proofs are carried out by simulating a *two-counter automaton (**tca**)* $\mathcal{A}$ (cf. [11]) and a reduction to the *halting problem on empty input* (we write $\mathcal{A}\downarrow$ for '$\mathcal{A}$ halts on empty input'). A **tca** is essentially a (nondeterministic) push-down automaton with two stacks and exactly two stack symbols (one of them is the initial stack symbol). This kind of machines has the same computation power as Turing machines.

Definition 9 (Two-counter automaton (cf. [11])) *A **tca** $\mathcal{A}$ is given by $(S, \Gamma, s^{init}, S_f, \Delta)$ where S is a finite set of states, Γ is the finite input alphabet, $s^{init} \in S$ is the initial state, $S_f \subseteq S$ is the set of final states, and $\Delta \subseteq (S \times \Gamma \times \{0,1\}^2) \times (S \times \{-1,1\}^2)$ is the transition relation such that if $((s,a,E_1,E_2),(s',C_1,C_2)) \in \Delta$ and $E_i = 0$ then $C_i \neq -1$ for $i = 1,2$ (to ensure that an empty counter cannot further be decremented). In the case of an empty input, we ignore the alphabet and assume $\Delta \subseteq (S \times \{0,1\}^2) \times (S \times \{-1,1\}^2)$.*

A **tca** effectively is a transition system equipped with two counters that influence the transitions. Each transition step of the automaton may rely on any of the counters being zero or non-zero and in each step the counters can be incremented or decremented. It is important to note that a **tca** can only distinguish between a counter being zero or non-zero. Consider the transition $((s,E_1,E_2),(s',C_1,C_2)) \in \Delta$. Here, $E_i = 1$ (resp. $= 0$) represents that counter i is non-empty (resp. empty) and $C_k = 1$ (resp. $= -1$) denotes that counter i is incremented (resp. decremented) by 1. The transition encodes that in state s the automaton can change its state to s' provided that the first (resp. second) counter meets condition E_1 (resp. E_2). The value of counter k changes according to C_k for $k = 1,2$. The transition $((s,1,0),(s',-1,1) \in \Delta$, for example, is enabled if the current sate is s, counter 1 is non-empty, and counter 2 is empty. If the transition is selected the state changes to s', counter 1 is decremented and counter 2 is incremented by 1.

The general mode of operation is as for pushdown automata. In particular, a *configuration* is a triple $(s, v_1, v_2) \in S \times \mathbb{N}_0^2$ describing the current state (s), the value of counter 1 (v_1) and of counter 2 (v_2). A *computation* δ a sequence of subsequent configurations that can emerge by transitions according to Δ such that the first state is s^{init}. An *accepting* configuration is a finite computation $\delta = (s_i, v_1^i, v_2^i)_{i=1,\dots,k}$ where the last state $s_k \in S_f$, i.e., a final state. We use $\delta_i = ((s_i, E_1^i, E_2^i),(s_{i+1}, C_1^i, C_2^i)) \in \Delta$ to denote the

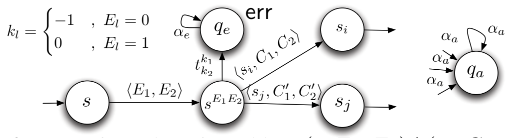

Figure 2. Transformation of transitions $(s,E_1,E_2)\Delta(s_i,C_1,C_2)$ and $(s,E_1,E_2)\Delta(s_j,C_1',C_2')$.

tuple that leads from the ith configuration (s_i, v_1^i, v_2^i) to the $i+1$th configuration $(s_{i+1}, v_1^{i+1}, v_2^{i+1})$ for $i < k$. In particular, we have that $v_j^{i+1} = v_j^i + C_j^i$ for $j = 1,2$.

Idea for the Reduction. In order to show that model checking of resource-bounded agent logics is undecidable, we reduce the halting problem to these logics. The specific construction varies for each logic. In the following we present the general idea. Detailed proofs can be found in [6]. Let $\mathcal{A} = (S, \Gamma, s^{\text{init}}, S_f, \Delta)$ be a **tca**. We represent the value of the two counters as resource types R_1 and R_2. For each state of the automaton, we add a state to the model and we label the accepting states in S_f by a proposition halt. The increment and decrement of counter values are modeled by actions producing and consuming from the corresponding resource type. The general idea underlying all the reductions is as follows (the path formula depends on the specific logic **L** considered):

($\star$) *$\mathcal{A}\downarrow$ iff there is a path in the **RBM** along which a path formula $\gamma_{\mathbf{L}}$ is true.*

The path in the **RBM** corresponds to an accepting computation of the automaton. The general mode of operation is straightforward and only the following difficulty remains: it is *not* possible to test whether a counter (i.e. a resource type) is empty in *any* of the resource-bounded agent logics. This causes problems in the reductions. For example, consider a tuple $((s,1,0),(s',-1,1) \in \Delta$. It can only be chosen if the second counter is actually empty. But, because we cannot directly test whether a resource type is empty, we need to come up with a workaround. This is the sophisticated part in the reductions (sometimes easier sometimes harder, depending on the expressiveness of the used logic). Fundamentally, the encoding of a transition $r := ((s,E_1,E_2),(s',C_1,C_2))$ is a three-step process (cf. Figure 2). In a state s of the **RBM** (we are economic and use the same notation) an agent performs an action $\langle E_1, E_2\rangle$ in order to 'select' r resulting in a 'test' state $s^{E_1E_2}$. In this state, an action $\langle s', C_1, C_2\rangle$ with resource-costs corresponding to the values of C_i can be executed (i.e. the action produces/consumes C_i resources of R_i). Clearly, such an action is only successful if sufficient resources are available. The check whether a counter/resource type is empty or not, happens at the intermediate state $s^{E_1E_2}$. In these states, a *non-cost-free* action $t_{k_2}^{k_1}$ for $k_i \in \{0,-1,1\}$ leading to an 'error state' q_e is available. Thus, if a counter should be zero according to the transition t; then, such a test action must not be performable. Hence, ($\star$) can be refined to the following:

($\star\star$) *$\mathcal{A}\downarrow$ iff there is path in the **RBM** such that eventually halt and along which there is no way to reach the error state q_e.*

Intuitively, if the error state cannot be reached along a path the selection of transitions is valid in the sense described above (i.e. it corressponds to an accepting computation of the automaton).

Non-flat Languages. We begin with specialised settings for non-flat languages. In the first case of **RAL**$_r$ we test whether there is a path such that eventually halt and in no state a transition to err is possible. In order to test whether the error state can be reached we make use of the non-resource flatness of the logic. Formally, we show: $\mathcal{A}\downarrow$ iff $\mathfrak{M}^{\mathcal{A}}, s^{\text{init}}, \eta_0 \models_r \neg\langle\langle\emptyset\rangle\rangle^{\eta_0}_{\mathbb{A}\text{gt}}\neg((\neg\langle\langle\emptyset\rangle\rangle\bigcirc\neg\mathsf{err})\,\mathcal{U}\,\mathsf{halt})$. The endowment η_0 equips agents with no resources.

Theorem 2 *Model checking* $\boldsymbol{RAL}_r$ *is undecidable, even in the single agent case; hence also,* $\boldsymbol{RAL}^+_r$ *and* $\boldsymbol{RAL}^*_r$ *are undecidable.*

Proof. [Sketch] Given a **tca** $\mathcal{A} = (S, \Gamma, s^{\text{init}}, S_f, \Delta)$ we construct an **RBM** $\mathfrak{M}^{\mathcal{A}}$ with two resources R_1 and R_2 (one per counter). We set $Q_{\mathfrak{M}^{\mathcal{A}}} = S \cup \{s^{E_1E_2} \mid s \in S, E_1, E_2 \in \{0,1\}\} \cup \{q_e, q_a\}$. State q_e (resp. q_a) is labelled err (resp. halt) and represents the 'error' (resp. 'halting') state. The states $s^{E_1E_2}$ are temporary states encoding that counter k is zero ($E_k = 0$) or non-zero ($E_k = 1$) for $k = 1, 2$.

For each transition $(s, E_1, E_2)\Delta(s', C_1, C_2)$ of the automaton we introduce actions $\langle E_1, E_2\rangle$ and $\langle s', C_1, C_2\rangle$ (cf. Figure 2). The first action leads from s to $s^{E_1E_2}$ and the second action from $s^{E_1E_2}$ to s'. Action $\langle s', C_1, C_2\rangle$ consumes/produces C_i units of resource R_i, $i = 1, 2$. The other kinds of actions are cost-free. Clearly, actions can only be performed if sufficient resources are available. We need to ensure that actions $\langle E_1, E_2\rangle$ with some $E_i = 0$ can only be performed if the counter i is actually 0; that is, if *no* resources of type R_i are available. Therefore, special 'test' actions $t^{k_1}_{k_2}$ that cost k_i units of resource R_i are introduced, $k_i \in \{0, -1, 1\}$. Such actions can only be performed in states $s^{E_1E_2}$ with some $E_i = 0$ and they always lead to state q_e. Now, in a sate $s^{E_1E_2}$ with some element equal 0, say $E_1 = 0$, $E_2 = 1$, (representing that counter 1 should be zero and 2 be non-zero) action t^{-1}_0 can be used to verify whether the currently available resources model the counter correctly: If q_e is reachable resources of type R_1 are available although this should not be the case according to E_1. Moreover, we add an action α_e to state q_e, leading back to q_e and an action α_a that leads from any state $s \in S_f$ to q_a and from q_a to itself. We assume that these are the only actions in states q_e and q_a and that they will be executed by default.
We show: $\mathcal{A}\downarrow$iff $\mathfrak{M}^{\mathcal{A}}, s^{\text{init}}, \eta_0 \models_r \neg\langle\langle\emptyset\rangle\rangle^{\eta_0}_{\mathbb{A}\text{gt}}\neg((\neg\langle\langle\emptyset\rangle\rangle\bigcirc\neg\mathsf{err})\,\mathcal{U}\,\mathsf{halt})$. Again, the formula states that there is an $(\eta_0, \epsilon, \mathbb{A}\text{gt})$-path such that eventually halt and the error state can never be reached along the way to q_a.
"$\Rightarrow$": Let $\delta = (s_i, v^i_1, v^i_2)_{i=1,\dots,k}$ be an accepting configuration. Clearly, if agent 1 executes $\langle E^i_1, E^i_2\rangle$ in state $s_i \notin S_f$, action $\langle s_{i+1}, C^i_1, C^i_2\rangle$ in state $s_i^{E^i_1E^i_2}$ (according to δ_i as introduced above), and α_a in $s_k \in S_f$ the resulting path is given by λ with $\lambda|_Q = (s_j s_j^{E^j_1E^j_2} s_{j+1})_{j=1,\dots,k-1}(q_a)^\omega$. It remains to show that for any state $s_i^{E^i_1E^i_2}$ with $E^i_k = 0$ we have that $\lambda|_{\mathcal{R}es}[2i-1](1, R_k) = 0$ (i.e. in this state agent 1 has no resources of type R_k). By induction one can easily prove that the actions keep track of the resources correctly and thus action t^{-1}_0 cannot be executed in any $s_j^{E^j_1E^j_2}$ along the path.
"$\Leftarrow$": Clearly, if such a satisfying path exists it must have the structure as shown above and we can directly construct an accepting computation of the automaton. Each triple $s_i s_i^{E^i_1E^i_2} s_{i+1}$ uniquely determines a transition δ_i. ■

In the previous case it was essential to keep track of the resources of the opponent. Here, we show that also the proponent-restricted setting is undecidable if we allow perfect-recall strategies. A perfect-recall strategy of the *proponent* is used to *encode* the computation of the automaton. Similar to Theorem 2, we can utilise the following reduction: $\mathcal{A}\downarrow$ iff $\mathfrak{M}^{\mathcal{A}}, s^{\text{init}}, \eta_0 \models_R \langle\langle 1\rangle\rangle^{\eta_0}((\neg\langle\langle 1\rangle\rangle\bigcirc\mathsf{err})\,\mathcal{U}\,\mathsf{halt})$.

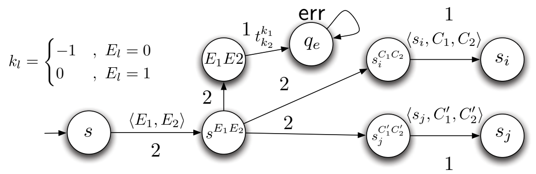

Figure 3. Construction used in the proof of Theorem 4 for $(s, E_1, E_2)\Delta(s_i, C_1, C_2)$ and $(s, E_1, E_2)\Delta(s_j, C'_1, C'_2)$.

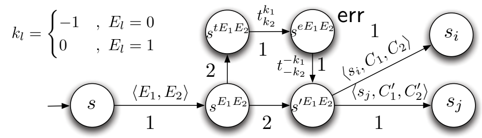

Figure 4. Construction used in the proof of Theorem 6 for $(s, E_1, E_2)\Delta(s_i, C_1, C_2)$ and $(s, E_1, E_2)\Delta(s_j, C'_1, C'_2)$.

Theorem 3 *Model checking pr-*$\boldsymbol{RAL}_R$ *(even without the release operator) is undecidable in the single-agent case; hence, also pr-*$\boldsymbol{RAL}^+_R$*, pr-*$\boldsymbol{RAL}^*_R$*,* $\boldsymbol{RAL}_R$*,* $\boldsymbol{RAL}^+_R$*, and* $\boldsymbol{RAL}^*_R$ *are undecidable.*

For the next setting, the proponent has once again no memory available. In turn, an additional agent (opponent agent 2) is used to model the computation (as in Theorem 2) and the proponent (agent 1) keeps track of the resources (as in Theorem 3). Note that it is important for the language *not* to be resource-flat. The idea of the construction is shown in Figure 3. Then, we can show that $\mathcal{A}\downarrow$ iff $\mathfrak{M}^{\mathcal{A}}, s^{\text{init}}, \eta_0 \models_r \neg\langle\langle 1\rangle\rangle^{\eta_0}\neg((\neg\langle\langle 2\rangle\rangle\bigcirc\langle\langle 1\rangle\rangle\bigcirc\mathsf{err})\,\mathcal{U}\,\mathsf{halt})$.

Theorem 4 *Model checking pr-*$\boldsymbol{RAL}_r$ *is undecidable for models with at least two agents; hence, also pr-*$\boldsymbol{RAL}^+_r$ *and pr-*$\boldsymbol{RAL}^*_r$ *are undecidable.*

Flat Languages. Resource-flat logics seem easier to verify as in the reduction it is not possible to have nested operators in order to verify whether the resources in a state are actually zero (compare the techniques introduced in the above). We show that perfect-recall and two agents can be used to 'overcome this limitation'. The proponent (agent 1) is used to simulate the computation of the automaton where the opponent (agent 2) tries to enter the error state in each test state; hence, no nested cooperation modality is needed. The setting is as shown in Figure 3 and we can show: $\mathcal{A}\downarrow$ iff $\mathfrak{M}^{\mathcal{A}}, s^{\text{init}}, \eta_0 \models_R \langle\langle 1\rangle\rangle^{\eta_0}_{\mathbb{A}\text{gt}}\Diamond\,\mathsf{halt}$.

Theorem 5 *Model Checking rf-*$\boldsymbol{RAL}_R$ *is undecidable for models with at least two agents; thus, also rf-*$\boldsymbol{RAL}^+_R$ *and rf-*$\boldsymbol{RAL}^*_r$ *are undecidable.*

At present, the decidability of the resource-flat *and* proponent-restricted versions of $\mathcal{L}_{\text{RAL}^+}$ and $\mathcal{L}_{\text{RAL}}$ with the standard semantics is open. However, by using the apparently stronger infinity-semantics ($\models^\infty_R$) we can prove the undecidability of *rf-pr-*$\mathcal{L}_{\text{RAL}}$ and thus also of *rf-pr-*$\boldsymbol{RAL}^*_R$ by Proposition 1. We do this by showing $\mathcal{A}\downarrow$ iff $\mathfrak{M}^{\mathcal{A}}, s^{\text{init}}, \eta_0 \models^\infty_R \langle\langle 1\rangle\rangle^{\eta_0}(\neg\mathsf{err})\,\mathcal{U}\,\mathsf{halt}$. The construction is sketched in Figure 4. Essentially, the opponent (2) may decide to enter the 'test *loop*' in $s^{E_1E_2}$. This 'bad' *loop* can only be avoided if 1 chooses good transitions of the automaton. Finite dead-end paths are disregarded thanks to the infinity-semantics.

Theorem 6 *Model Checking rf-pr-*$\boldsymbol{RAL}^*_R$*, rf-pr-(*$\mathcal{L}_{RAL}$*,*$\models^\infty_R$*), and rf-pr-(*$\mathcal{L}_{RAL}$*,*$\models^\infty_R$*) is undecidable for models with at least two agents.*

	$\mathcal{L}_{\mathrm{RAL}^*}$	$\mathcal{L}_{\mathrm{RAL}^+}$	$\mathcal{L}_{\mathrm{RAL}}$	pr-$\mathcal{L}_{\mathrm{RAL}^*}$	pr-$\mathcal{L}_{\mathrm{RAL}^+}$	pr-$\mathcal{L}_{\mathrm{RAL}}$
$\models_R$	U^1	U^1	U^1	U^1	U^1	U^1
$\models_r$	U^1	U^1	U^1	U^2	U^2	U^2
rf+$\models_R$ / $\models_R^\infty$	U^2	U^2	U^2	U^2/ U^2_∞	**?** / U^2_∞	**?** / U^2_∞
rf+$\models_r$	**?**	**?**	**?**	**?**	**?**	**?**
$\models_R^k, \models_r^k$	**D**	**D**	**D**	**D**	**D**	**D**

Table 1. Overview of model-checking decidability results. Each cell represents the logic over the language given in the column using the semantics given in the row. The content of each cell indicates whether the model-checking problem is decidable (D) or undecidable (U^x). x indicates the number of required agents. U^2_∞ refers to the semantics $\models_R^\infty$.

Summary of the Complexity Results. Our analysis, summarised in Table 1, shows that the combination of various settings and languages influences the difficulty of the model-checking problem. Although we do not claim that our results with respect to the number of agents are optimal they show an interesting pattern. One can often compensate the lack of expressiveness caused by various restrictions on the language or semantics by taking more agents into account. The most difficult case seems to be the perfect-recall semantics where resource-flatness suggests to be important for decidable fragments, particularly in combination with memoryless strategies.

The question for the resource-flat proponent-restricted languages $\mathcal{L}_{\mathrm{RAL}^+}$ and $\mathcal{L}_{\mathrm{RAL}}$ under the R-semantics is *still open*, while the case is proven undecidable if focusing on infinite paths. Also open is the case of resource-flat languages with r-semantics. The two bounded settings are decidable.

Note, that the result form [5] about the decidability of **RTL** matches the results presented here, since it corresponds to the single-agent case of rf-pr-$\mathbf{RAL}_R$.

4 Related Work & Conclusions

Related Work. *Resource-Bounded Tree Logics*, introduced in [5], extend the well-known Computation Tree Logics [10] by resources. Instead of asking for the plain existence of an infinite path satisfying some temporal property, this path must also be feasible given a set of available resources. As shown in [6] these logics can be considered as the resource-flat-single-agent fragments of the logics presented here.

Resource-Bounded Coalition Logic (**RBCL**), an extension of Coalition Logic with resources, is introduced [2]. This logic can be seen as a first step towards a multi-agent extension of the Resource-Bounded Tree Logics [5] under the restricted temporal setting of multiple-step strategies ('sometime in the future'). Only recently, in [3] a multi-agent version (**RBATL**) following the same ideas is presented. For both logics the authors allow only consumption of resources which is computationally much easier and has a decidable model-checking property (cf. Theorem 1). In [6] we show that **RBCL** can essentially be embedded in $\mathbf{RAL}_R$; the same seems possible for **RBATL**.

RBCL is used in [1] to specify and verify properties about *Coalitional Resource Games* [16]. These are games in which agents can cooperate and combine their available resources in order to bring about desired goals.

Conclusions. We have presented various strategic logics for reasoning about abilities under limited resources. The different settings were based on classical restrictions (cf. [10, 4]) imposed on the underlying temporal language ($\mathcal{L}_{\mathrm{RAL}^*}$ vs. $\mathcal{L}_{\mathrm{RAL}^+}$ vs. $\mathcal{L}_{\mathrm{RAL}}$) and strategic dimension (perfect vs. imperfect recall). Additionally, we have imposed restrictions on the resource dimension by focussing on specific groups acting under limited resources (proponent-restrictiveness) and on the nesting of cooperation operators (resource-flatness).

Our main objective was to analyse whether it is possible to verify resource-bounded agents under these diverse settings. We have shown undecidability for many fragments and identified the number of agents needed. We believe that these results are important and interesting for future investigations of strategic abilities under limited resources. Our results show that small changes in the language and semantics may influence whether model checking becomes decidable or undecidable (cf. for instance, the $\models_r^\infty$ and $\models_r$ semantics over rf-pr-$\mathcal{L}_{\mathrm{RAL}}$). We have also considered bounded settings with decidable model-checking problems.

For future work, we plan to close the open cases, in particular for the resource-flat languages under r-semantics and to analyse the model-checking complexity of the decidable and tractable fragments. An extended version of this paper containing full proofs can be found in [6].

REFERENCES

[1] N. Alechina, B. Logan, N. Hoang Nga, and A. Rakib, 'Verifying properties of coalitional ability under resource bounds', in *Proceedings of the Second Internatinoal Workshop on Logics for Agents and Mobility (LAM'09)*, ed., Berndt Farwer, Los Angeles CA, USA, (August 2009).

[2] N. Alechina, B. Logan, N. Hoang Nga, and A. Rakib, 'A logic for coalitions with bounded resources', in *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence*, ed., Craig Boutilier, volume 2, pp. 659–664, Pasadena CA, USA, (July 2009). IJCAI/AAAI, AAAI Press.

[3] N. Alechina, B. Logan, N. Hoang Nga, and A. Rakib, 'Resource-bounded alternating-time temporal logic', in *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, eds., Wiebe van der Hoek, Gal Kaminka, Yves Lespérance, Michael Luck, and Sandip Sen, Toronto, Canada, (May 2010). IFAAMAS. (to appear).

[4] R. Alur, T. A. Henzinger, and O. Kupferman, 'Alternating-time Temporal Logic', *Journal of the ACM*, **49**, 672–713, (2002).

[5] N. Bulling and B. Farwer, 'Expressing properties of resource-bounded systems: The logics RBTL and RBTL*', in *Post-Proceedings of CLIMA '09*, eds., J. Dix, Michael Fisher, and Peter Novak, Hamburg, Germany, (September to appear 2010).

[6] N. Bulling and B. Farwer, 'On the decidability of verifying resource-bounded agents', Technical Report IfI-10-05, Clausthal University of Technology, (Mai 2010).

[7] N. Bulling and W. Jamroga, 'What agents can probably enforce', *Fundamenta Informaticae*, **93**, 81–96, (2009).

[8] N. Bulling, W. Jamroga, and J. Dix, 'Reasoning about temporal properties of rational play', *Annals of Mathematics and Artificial Intelligence*, **53**(1-4), 51–114, (2009).

[9] E.M. Clarke, O. Grumberg, and D. Peled, *Model Checking*, MIT Press, 1999.

[10] E.M. Clarke and E.A. Emerson, 'Design and synthesis of synchronization skeletons using branching time temporal logic', in *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pp. 52–71, (1981).

[11] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachusetts, 1979.

[12] W. Jamroga and J. Dix, 'Model checking abilities of agents: A closer look', *Theory of Computing Systems*, **42**(3), 366–410, (2008).

[13] O. Kupferman, M.Y. Vardi, and P. Wolper, 'An automata-theoretic approach to branching-time model checking', *Journal of the ACM*, **47**(2), 312–360, (2000).

[14] F. Laroussinie, N. Markey, and G. Oreiby, 'On the expressiveness and complexity of atl', *CoRR*, **abs/0804.2435**, (2008).

[15] A. Pnueli and R. Rosner, 'On the synthesis of a reactive module', in *POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 179–190, New York, NY, USA, (1989). ACM.

[16] M. Wooldridge and P.E. Dunne, 'On the computational complexity of coalitional resource games', *Artif. Intell.*, **170**(10), 835–871, (2006).