On the Boundary of (Un)decidability: Decidable Model-Checking for a Fragment of Resource Agent Logic*

Natasha Alechina[1] and **Nils Bulling**[2] and **Brian Logan**[1] and **Hoang Nga Nguyen**[1]
[1] School of Computer Science, University of Nottingham, UK
[2] Delft University of Technology, The Netherlands
[1] {nza,bsl,hnn}@cs.nott.ac.uk, [2] n.bulling@tudelft.nl

Abstract

The model-checking problem for Resource Agent Logic is known to be undecidable. We review existing (un)decidability results and identify a significant fragment of the logic for which model checking is decidable. We discuss aspects which makes model checking decidable and prove undecidability of two open fragments over a class of models in which agents always have a choice of doing nothing.

1 Introduction

There exist several formalisms that extend Alternating Time Temporal Logic (ATL), [Alur *et al.*, 2002] with reasoning about resources available to agents and production and consumption of resources by actions, see, for example, [Bulling and Farwer, 2010; Alechina *et al.*, 2010; Della Monica *et al.*, 2013; Alechina *et al.*, 2014; Bulling and Goranko, 2013]. When the production of resources is allowed, the model-checking problem for many of these logics is undecidable [Bulling and Farwer, 2010; Bulling and Goranko, 2013]. Recently, however, it was shown that some resource logics with production of resources have a decidable model checking problem [Alechina *et al.*, 2014]. In this paper, we investigate the reasons for decidability or undecidability of the model-checking problem for resource logics. There is a quite bewildering variety of choices for the syntax and semantics of resource logics, for example, the precise definition of when a joint action is 'affordable' by a coalition of agents (can the agents pool their resources, and can they use resources produced by the joint action to offset the costs involved in the action). Many of these choices do not affect the decidability or otherwise of the model-checking problem. In particular, the decidability result of [Alechina *et al.*, 2014] was proven in the presence of two major restrictions called, using the notion of [Bulling and Farwer, 2010], *resource flat* and *proponent restricted*. The former assumes that agents are always re-equipped with fresh resources when they reconsider their strategies; the latter assumes that only the proponents act under resource bounds. In addition to these restrictions, another choice in the semantics seems to be relevant for decidability. This choice, which is also related to the finitary and infinitary semantics of [Bulling and Farwer, 2010], stipulates whether in every model, agents always have a choice of doing nothing (executing an *idle action*) that produces and consumes no resources [Alechina *et al.*, 2014]. Apart from the technical convenience for model-checking (intuitively it implies that any strategy to satisfy a next or until formula only needs to ensure the relevant subformula becomes true after finitely many steps, and after that the agents can always choose the idle action which does not increase the 'cost' of the strategy), this choice is motivated in [Alechina *et al.*, 2010] by the properties of the logic, e.g., coalition monotonicity.

In this paper, we investigate the effects of various semantic choices such as idle on the decidability of the model-checking problem. First we show that the resource-flat as well as the proponent-restricted fragment of resource agent logic remain undecidable in the presence of idle actions. We then identify and motivate a significant, non resource-flat fragment that has a decidable model checking property in the presence of idle actions, and is not decidable otherwise. It follows that idle actions can make a difference for the decidability of model checking with respect to the semantics considered in this paper.

The paper is organised as follows. In Section 2 we introduce our version of resource agent logic, its models and the semantics. Afterwards, we put our setting in context with existing work, discussing the main differences and the main (un)decidability results. Based on this comparison we review results shedding light into different semantic choices. We give our first technical undecidability results and identify a new, non resource-flat fragment of resource agent logic. Section 4 presents our second technical result: a decidability result for the newly identified fragment. We conclude in Section 5.

2 Resource Agent Logic

In this section we define the logic *resource agent logic* RAL and resource-bounded models. We essentially follow [Bulling and Farwer, 2010], combined with aspects from [Alechina *et al.*, 2014]. We point out the similarities and differences in more detail in Section 3.1.

Syntax. The logic is defined over a set of agents $\mathbb{A}\mathrm{gt}$, a set of resources types Res and a set of propositional symbols Π.

*Acknowledgements: This work was supported by the Engineering and Physical Sciences Research Council [grant EP/K033905/1].

An *endowment (function)* $\eta : \mathbb{A}\mathrm{gt} \times Res \rightarrow \mathbb{N}_0^{\infty}$ is used to assign resources to agents; $\eta_a(r) = \eta(a, r)$ is the number of resources agent a has of resource type r. With En we denote the set of all possible endowments. Resource types can represent, for example, money, fuel, and battery power. Special minimal and maximal endowment functions are denoted by $\bar{0}$ and $\bar{\infty}$, respectively. The former expresses that there are no resources at all, whereas the latter equips all agents with an infinite amounts of resources. The logic RAL is defined according to the grammar of ATL [Alur *et al.*, 2002] where two types of cooperation modalities are available, the meaning of which is explained below. RAL-formulae are defined by:

$$\phi ::= \mathsf{p} \mid \neg\phi \mid \phi \wedge \phi \mid \langle\!\langle A\rangle\!\rangle_B^{\downarrow}\mathbf{X}\varphi \mid \langle\!\langle A\rangle\!\rangle_B^{\eta}\mathbf{X}\varphi \mid \langle\!\langle A\rangle\!\rangle_B^{\downarrow}\varphi\mathbf{U}\psi \mid \langle\!\langle A\rangle\!\rangle_B^{\eta}\varphi\mathbf{U}\psi \mid \langle\!\langle A\rangle\!\rangle_B^{\downarrow}\mathbf{G}\varphi \mid \langle\!\langle A\rangle\!\rangle_B^{\eta}\mathbf{G}\varphi$$

where $\mathsf{p} \in \Pi$ is a proposition, $A, B \subseteq \mathbb{A}\mathrm{gt}$ are sets of agents, and η is an endowment. We also define $\langle\!\langle A\rangle\!\rangle^{\downarrow}$ and $\langle\!\langle A\rangle\!\rangle^{\eta}$ as abbreviations for $\langle\!\langle A\rangle\!\rangle_A^{\downarrow}$ and $\langle\!\langle A\rangle\!\rangle_A^{\eta}$, respectively. The operators $\mathbf{X}$, $\mathbf{U}$, and $\mathbf{G}$ denote the standard temporal operators expressing that some property holds in the *next* point in time, *until* some other property holds, and *now and always* in the future, respectively. Both types of cooperation modality, $\langle\!\langle A\rangle\!\rangle_B^{\downarrow}$ and $\langle\!\langle A\rangle\!\rangle_B^{\eta}$, assume that *all* agents in $A \cup B$ consume and produce resources. The reading of $\langle\!\langle A\rangle\!\rangle_B^{\eta}\varphi$ is that *agents A have a strategy compatible with the endowment η* to enforce φ. The formula $\langle\!\langle A\rangle\!\rangle_B^{\downarrow}\varphi$ reads similarly but the strategy must be compatible with the resources currently available to the agents. In both cases compatible means that the strategy can be executed given the agents' resources. For both modalities it is necessary to keep track of resource production and consumption during the execution of a strategy. The evaluation of a modality $\langle\!\langle A\rangle\!\rangle_B^{\eta}$, however, (re-)equips all agents with a *fresh* amount of resources: the current resource endowment is overwritten by endowment η.

Semantics. We define the models of RAL as in [Bulling and Farwer, 2010]. Following [Alechina *et al.*, 2014] we also define a special case of these models in which agents have an *idle action* in their repertoire which neither consumes nor produces resources.

Definition 1 (RBM, iRBM). *A resource-bounded model (*$\mathbf{RBM}$*) is given by* $\mathfrak{M} = (\mathbb{A}\mathrm{gt}, Q, \Pi, \pi, Act, d, o, Res, t)$ *where* $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$ *is a set of agents;* $\pi : Q \rightarrow \wp\Pi$ *is a valuation of propositions; Act is a finite set of actions; and the function* $d : \mathbb{A}\mathrm{gt} \times Q \rightarrow \wp Act \backslash \{\emptyset\}$ *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* $d_1(q) \times \ldots \times d_k(q)$ *of action profiles in state* q*. Similarly,* $d_A(q)$ *denotes the action tuples available to* A *in* q*.* o *is a serial transition function which maps each state* $q \in Q$ *and action profile* $\alpha = \langle\sigma_1, \ldots, \sigma_k\rangle \in d(q)$ *(specifying a move for each agent) to another state* $q' = o(q, \alpha)$*. Finally, the function* $t : Act \times Res \rightarrow \mathbb{Z}$ *models the resources consumed and produced by actions. We define* $\mathsf{prod}(\sigma, r) := \max\{0, t(\sigma, r)\}$ *(resp.* $\mathsf{cons}(\sigma, r) := \min\{0, t(\sigma, r)\}$*) as the amount of resource* r *produced (resp. consumed) by action* σ*. For* $\alpha = \langle\sigma, \ldots, \sigma_k\rangle$*, we use* α_A *to denote the sub-tuple consisting of the actions of agents* $A \subseteq \mathbb{A}\mathrm{gt}$*.*

An $\mathbf{RBM}$ with idle actions*,* $\mathbf{iRBM}$ *for short, is an* $\mathbf{RBM}$ $\mathfrak{M}$ *such that for all agents* a*, all states* q*, and all resource types* r *in* $\mathfrak{M}$*, there is an action* $\sigma \in d_a(q)$ *with* $t(\sigma, r) = 0$*. We refer to this action (or to one of them if there is more than one) as the* idle action *of* a *and denote it by* $idle$*.*

A *path* $\lambda \in Q^{\omega}$ is an infinite sequence of states such that there is a transition between two adjacent states. A *resource-extended* path $\lambda \in (Q \times \mathsf{En})^{\omega}$ is an infinite sequence over $Q \times \mathsf{En}$ such that the restriction to states (the first component), denoted by $\lambda|_Q$, is a path in the underlying model. The projection of λ to the second component of each element in the sequence is denoted by $\lambda|_{\mathsf{En}}$. We define $\lambda[i]$ to be the $i+1$-th state of λ, and $\lambda[i, \infty]$ to be the suffix $\lambda[i]\lambda[i+1]\ldots$. A *strategy* for a coalition $A \subseteq \mathbb{A}\mathrm{gt}$ is a function $s_A : Q^+ \rightarrow Act^A$ such that $s_A(\lambda q) \in d_A(q)$ for $\lambda q \in Q^+$. Such a strategy gives rise to a set of (resource-extended) paths that can emerge if agents follow their strategies. A *(η, s_A, B)-path* is a resource-extended path λ such that for all $i = 0, 1, \ldots$ with $\lambda[i] := (q_i, \eta^i)$ there is an action profile $\alpha \in d(\lambda|_Q[i])$ such that: (1) $\eta^0 = \eta$ (η describes the initial resource distribution), (2) $s_A(\lambda|_Q[0, i]) = \alpha_A$ (A follow their strategy), (3) $\lambda|_Q[i+1] = o(\lambda|_Q[i], \alpha)$ (transition according to α), (4) for all $a \in A \cup B$ and $r \in Res$: $\eta_a^i(r) \geq \mathsf{cons}(\alpha_a, r)$ (each agent has enough resources to perform its action), (5) for all $a \in A \cup B$ and $r \in Res$: $\eta_a^{i+1}(r) = \eta_a^i(r) + \mathsf{prod}(\alpha_a, r) - \mathsf{cons}(\alpha_a, r)$ (resources are updated). (6) for all $a \in \mathbb{A}\mathrm{gt} \setminus (A \cup B)$ and $r \in Res$: $\eta_a^{i+1}(r) = \eta_a^i(r)$ (resources remain unchanged for agents not in $A \cup B$). The *(η, B)-outcome* of a strategy s_A in q, $out(q, \eta, s_A, B)$, is defined as the set of all (η, s_A, B)-paths starting in q. Truth is defined over an $\mathbf{RBM}$ $\mathfrak{M}$, a state $q \in Q_{\mathfrak{M}}$, and an endowment η. The *semantics* is given by the satisfaction relation $\models$ where the cases for propositions, negation and conjunction are standard and omitted:

$\mathfrak{M}, q, \eta \models \langle\!\langle A\rangle\!\rangle_B^{\downarrow}\varphi$ iff there is a strategy s_A for A such that for all $\lambda \in out(q, \eta, s_A, B)$, $\mathfrak{M}, \lambda, \eta \models \varphi$

$\mathfrak{M}, q, \eta \models \langle\!\langle A\rangle\!\rangle_B^{\zeta}\varphi$ iff there is a strategy s_A for A such that for all $\lambda \in out(q, \zeta, s_A, B)$, $\mathfrak{M}, \lambda, \zeta \models \varphi$

$\mathfrak{M}, \lambda, \eta \models \mathbf{X}\varphi$ iff $\mathfrak{M}, \lambda|_Q[1], \lambda|_{\mathsf{En}}[1] \models \varphi$

$\mathfrak{M}, q, \eta \models \varphi\mathbf{U}\psi$ iff there exists i with $i \geq 0$ and $\mathfrak{M}, \lambda|_Q[i], \lambda|_{\mathsf{En}}[i] \models \varphi$ and for all j with $0 \leq j < i$ $\mathfrak{M}, \lambda|_Q[j], \lambda|_{\mathsf{En}}[j] \models \psi$

$\mathfrak{M}, q, \eta \models \mathbf{G}\varphi$ iff for all $i \geq 0$, $\mathfrak{M}, \lambda|_Q[i], \lambda|_{\mathsf{En}}[i] \models \varphi$

The *model checking problem* is stated as follows: does $\mathfrak{M}, q, \eta \models \varphi$ hold? When the context is clear, we simply write $q, \eta \models \varphi$; if φ is only a propositional formula, we might also ignore η.

Observe that the standard ATL modalities $\langle\!\langle A\rangle\!\rangle$ can be defined as $\langle\!\langle A\rangle\!\rangle_{\mathbb{A}\mathrm{gt}}^{\bar{\infty}}$, so the logic is a proper extension of ATL.

Remark 1. *We refer to the semantics introduced above as* infinitary semantics*. In [Bulling and Farwer, 2010] the main semantics also allows for finite (maximal) paths. We refer to that semantics as* finitary semantics*. We note that both semantics coincide over* $\mathbf{iRBM}$*s, as is is always possible to extend a path due to the idle actions.*

Syntactic fragments. Following [Bulling and Farwer, 2010] we define two fragments of the logic. The *resource-*

flat fragment, rfRAL, only allows cooperation modalities of type $\langle\langle A\rangle\rangle_B^\eta$: agents are always (re-)equipped with a fresh set of resources whenever they re-consider their strategies. The *proponent restricted fragment*, prRAL, only allows cooperation modalities of types $\langle\langle A\rangle\rangle^\downarrow$ and $\langle\langle A\rangle\rangle^\eta$: only the proponents are assumed to make use of resources. The fragment combining both restrictions is denoted by rfprRAL.

3 The Quest for Decidability

We first discuss differences and similarities to the original resource agent logics. We present the main idea underlying the undecidability of model-checking results from [Bulling and Farwer, 2010] and investigate the reasons for the (un)decidability. We present new undecidability results, and motivate a new fragment of RAL.

3.1 Two Related Logics

The logic presented here is based on the resource bounded logics of [Bulling and Farwer, 2010] and [Alechina *et al.*, 2014]. The authors of the former paper introduce a very general framework. Several interesting special cases are identified, and the (un)decidability of their model checking problems investigated. In the interests of readability, we refer to the setting of [Bulling and Farwer, 2010] by S_1 and to that of [Alechina *et al.*, 2014] by S_2. The language of RAL is almost identical to the setting of S_1, except that we do not allow the release operator. Essentially, S_2 corresponds to the resource-flat and proponent-restricted fragment of RAL. $\mathbf{RBM}$s serve as models of S_1, where S_2 uses $\mathbf{iRBM}$s. There are negligible differences in how the production and consumption of resources are handled. In S_2 both are considered at the same time; here, actions first consume resources and afterwards produce them. However this is just a modelling issue. Most results of S_1 are given in terms of the finitary semantics (cf. Remark 1). Finally, in S_1 agents belonging to the same group—either to the proponents or the opponents—are allowed to share their resources, whereas in our setting each agent has its own resource balance. This change does not affect the (un)decidability results given here, but eases the presentation.

3.2 Investigating the Boundary of (Un)Decidability

In [Bulling and Farwer, 2010] it was shown that many—actually most—fragments of their resource agent logic are undecidable. However, an interesting case remained open: the resource-flat, proponent-restricted fragment. Just recently, this open problem has been shown to be decidable in [Alechina *et al.*, 2014; Alechina *et al.*, 2015]:

Observation 1. rfprRAL *is decidable over* $\mathbf{iRBM}$*s.*

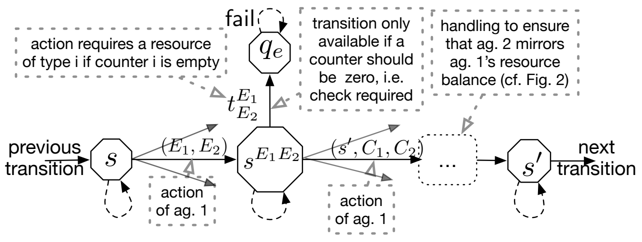
A natural question arises: *Could we extend the decidability to more expressive fragments*? We first show that proponent restrictedness is essential for decidability, including over $\mathbf{iRBM}$s.

The Non-Proponent Restricted Fragment

In [Bulling and Farwer, 2010] it was shown that model checking formulae of type $\langle\langle 1\rangle\rangle^{\bar{0}}_{\mathbb{A}\mathrm{gt}}\mathbf{F}\,\mathsf{halt}$ is undecidable over $\mathbf{RBM}$s. However the decidability of this fragment was open over $\mathbf{iRBM}$s. In Theorem 1, we show that undecidability continues to hold. Before we sketch the proof we present the basic idea underlying the reductions of [Bulling and Farwer, 2010].

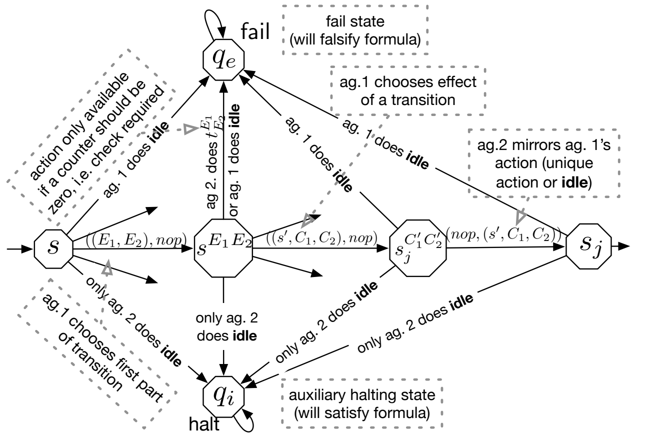
The undecidability of the model checking problems is shown by reducing the halting problem for *two counter machines* (also called Minsky machines) [Hopcroft and Ullman, 1979]. A two-counter machine is essentially a pushdown automaton with two stacks. The stacks are represented as two counters over natural numbers. Each of the two counters (1 and 2) can be incremented, decremented (if non-zero), and tested for zero. The behaviour of such an automaton is specified by a transition relation $(s, E_1, E_2)\Delta(s', C_1, C_2)$ with the following meaning: a transition from automaton state s to s' is *enabled* if counter $i \in \{1, 2\}$ satisfies condition $E_i \in \{\text{zero}, \text{non-zero}\}$ (with the obvious meaning). If the transition is *taken* the automaton changes its control state from s to s', and counter i is updated by adding $C_i \in \{-1, 0, +1\}$. It is said that the automaton halts on empty input iff the *halting state* of the automaton is reached by subsequently taking enabled transitions. The main idea of the reduction of the halting problem is to encode the transition table of the automaton as an $\mathbf{RBM}$. The two counters are simulated by two resource types. The model consists of two agents: the *simulator* agent 1 and the *spoiler* agent 2. Agent 1 is supposed to select transitions of the automaton where agent 2 is used to ensure that only enabled transitions are selected by agent 1. The basic modelling of a single transitions $(s, E_1, E_2)\Delta(s', C_1, C_2)$ is shown in Figure 1. The first action (E_1, E_2) of agent 1 is used to (partially) check whether a transition of the automaton is enabled. That is, if E_i = non-empty, agent 1 must have a resource of type i to execute the action. If such an action is taken, the system enters a "test state" $s^{E_1E_2}$. The purpose of the test state is to check whether a transition with E_i = empy is only selected by agent 1 if counter i is indeed zero, i.e. ensures a correct simulation. Note that, in general, nothing prevents agent 1 from executing such an action if it has resources available, although it should only be executed if no resources are available. Essentially, the problem is that it is not possible to directly *test for zero* in the model.[1] The workaround proposed in [Bulling and Farwer, 2010] is to use the spoiler agent 2 to perform the "zero test". The idea is that in test state $s^{E_1E_2}$, agent 2 must not be able to reach the *fail state* q_e. Reaching the fail state is only possible if resources are available when there should not be any. This is encoded by action $t^{E_1}_{E_2}$ in the model. For example, if counter 1 should be empty, E_1 = empty, the action $t^{E_1}_{E_2}$ can only be executed if resources of type 1 are available. However, this also requires that agent 2 correctly mirrors agent 1's resource balance, i.e. 2 also simulates the counter values. This is achieved by making the model turn-based. Once agent 1 has executed an action (s', C_1, C_2) an intermediate state is introduced in which agent 2 has a single choice with the same effect on the resources as agent 1's previous action (dotted rectangle in Fig. 1). Based on this encoding it is shown (using the finitary

[1]Testing for zero is a delicate property, the satisfaction of which seems crucial for the undecidability of other formalisms, such as Petri nets [Peterson, 1981].



fail
q_e
action requires a resource of type i if counter i is empty
transition only available if a counter should be zero, i.e. check required
handling to ensure that ag. 2 mirrors ag. 1's resource balance (cf. Fig. 2)
$t^{E_1}_{E_2}$
previous transition
s
(E_1, E_2)
$s^{E_1E_2}$
(s', C_1, C_2)
...
s'
next transition
action of ag. 1
action of ag. 1

Figure 1: The reduction used wrt. **RBM**s is without dashed loops, whereas for the reduction wrt. **iRBM**s (cf. Cor. 1) dashed loops have to be taken into account.



fail
q_e
fail state (will falsify formula)
action only available if a counter should be zero, i.e. check required
ag. 1 does idle
ag. 2 does $t^{E_1}_{E_2}$
or ag. 1 does idle
ag.1 chooses effect of a transition
ag. 1 does idle
ag. 1 does idle
ag.2 mirrors ag. 1's action (unique action or idle)
s
$((E_1, E_2), nop)$
$s^{E_1E_2}$
$((s', C_1, C_2), nop)$
$s_j^{C_1'C_2'}$
$(nop, (s', C_1, C_2))$
s_j
only ag. 2 does idle
ag.1 chooses first part of transition
only ag. 2 does idle
only ag. 2 does idle
only ag. 2 does idle
q_i
halt
auxiliary halting state (will satisfy formula)

Figure 2: The reduction used wrt. **iRBM**s for rfRAL.

semantics) that the automaton $\mathcal{A}$ halts on the empty input if, and only if, $\mathfrak{M}(\mathcal{A}), q_{\text{init}}, \bar{0} \models \langle\langle\{1\}\rangle\rangle^{\bar{0}}_{\{1,2\}}\mathbf{F}\,\mathsf{halt}$ where $\mathfrak{M}(\mathcal{A})$ is the **RBM** constructed from $\mathcal{A}$. The state corresponding to the automaton's accepting state is labelled halt. In order for the reduction to work over **iRBM**s the main difficulty is the correct mirroring of 1's resources by agent 2 in the presence of idle actions. The modified encoding of a transition $(s, E_1, E_2)\Delta(s', C_1, C_2)$ is illustrated in Figure 2.

Theorem 1. *Model checking* rfRAL *over the class of* **iRBM***s is undecidable, even for two agents and formulae of type* $\langle\langle 1\rangle\rangle^{\bar{0}}_{\{1,2\}}\boldsymbol{F}\mathsf{p}$.

Proof sketch. We argue that the automaton $\mathcal{A}$ halts on the empty input iff $\mathfrak{M}(\mathcal{A}), q_{\text{init}}, \bar{0} \models \langle\langle 1\rangle\rangle^{\bar{0}}_{\{1,2\}}\mathbf{F}\,\mathsf{halt}$, where $\mathfrak{M}$ is the encoding of the automaton following the idea shown in Figure 2. First, we observe that the execution of an idle action by agent 1 would immediately falsify the formula, as state q_e would be reached. Similarly, if agent 2 does idle and agent 1 does not, the formula will be true. As we are looking for a winning strategy of 1 against *all* strategies of 2 we can neglect the cases where any of the agents performs the idle action. Again, the encoding describes the modelling of a transition $(s, E_1, E_2)\Delta(s', C_1, C_2)$. In this modelling the simulation of agent 1's resources by agent 2 is made explicit. In states $s_j^{C_1'C_2'}$, agent 2 can only perform the same action (apart from the idle action) as the one selected by agent 1 in state $s^{E_1E_2}$ with the same resource consumption and production. We briefly sketch the correctness of the reduction. **(i)** $\mathcal{A}$ halts. Then, agent 1 simulates the transitions of the automaton's accepting run. Agent 2 will never be able to reach the fail state q_e. Moreover, either agent 2's resources correctly simulate 1's resources, or agent 2 does the idle action. In both cases either the halting state or the auxiliary halting state q_i, both labelled halt, are reached. The formula is true. **(ii)** Let the formula be true. Agent 1 must have a strategy that guarantees reaching a state labelled halt against all strategies of 2, including 2's strategy in which 2 never performs the idle action. This strategy of 2 correctly mirrors 1's resources and ensures that 1's strategy only selects enabled transitions. Thus, the strategy of 1 encodes an accepting run of the automaton. □

The Proponent Restricted Fragment

Theorem 1 shows that the restriction of resource-flatness is not enough to obtain a decidable model checking property. We turn to the proponent-restricted fragment. Below we show, by adopting the undecidability proof of [Bulling and Farwer, 2010] for prRAL to work over **iRBM**'s, that prRAL is also undecidable over **iRBM**s. This is a negative result; however, in contrast to Theorem 1, the formula used in the reduction is more complex, and leaves room for restricting the temporal structure of the language. Indeed, this is the motivation for the decidable fragment of prRAL that we introduce in Section 3.3.

Corollary 1 (of [Bulling and Farwer, 2010]). *Model checking* prRAL *over the class of* **iRBM***s is undecidable even in the case of a single agent.*

Proof sketch. The undecidability proof for prRAL over **RBM**s of [Bulling and Farwer, 2010] essentially follows the encoding shown in Figure 1. The difference is that the second agent is removed (as well as the dotted box, implementing the resource mirroring behaviour), and agent 1 itself is used to perform the "zero test". This requires a slightly more sophisticated formula: the automaton $\mathcal{A}$ halts on the empty input if, and only if, $\mathfrak{M}(\mathcal{A}), q_{\text{init}}, \bar{0} \models \langle\langle\{1\}\rangle\rangle^{\bar{0}}((\neg\langle\langle\{1\}\rangle\rangle^{\downarrow}\mathbf{X}\,\mathsf{fail})\mathbf{U}\,\mathsf{halt})$. The main idea is that in test state $s^{E_1E_2}$, agent 1 must not be able to reach the *fail state* q_e, which is expressed by $\neg\langle\langle\{1\}\rangle\rangle^{\downarrow}\mathbf{X}\,\mathsf{fail}$. Now, in order to extend the reduction to work over **iRBM**s, we add reflexive loops (dashed in the figure) which represent agent 1's idle actions. It is easy to see that the reduction still works. The agent would not be able to reach the halting state (labelled halt) if it had taken an idle loop forever, nor would it help the agent, in its role as opponent, to reach the fail state. □

3.3 A Decidable Fragment of RAL

Following the observation made in the previous section, we define a proponent-restricted but not resource-flat fragment of RAL that has a decidable model checking property. Let us first introduce the *positive fragment* of RAL as the set of all RAL-formulae where no cooperation modality is under the scope of a negation symbol, and let the ***U*-restricted** fragment of RAL restrict the use of the until operator $\mathbf{U}$ such that the formulae φ_1 on the left-hand-side of $\varphi_1\mathbf{U}\varphi_2$ is purely propositional.

Definition 2 (The fragment prRAL$^{\mathsf{r}}$). *The logic* prRAL$^{\mathsf{r}}$ *is defined as the proponent-restricted, positive and* **U***-restricted fragment of* RAL.

The prRAL$^{\mathsf{r}}$-fragment allows us to express properties of coalitions of agents which re-consider their strategies *without* being re-equiped with fresh resources. An example of a property expressible in prRAL$^{\mathsf{r}}$ (but not in a resource-flat fragment) is, for example, "given their initial battery charge, rescue robots A can safely get to a position from which they can perform rescue while in visual contact with the base". Formally, this can be specified by the formula $\langle\langle A\rangle\rangle^{\eta_{\text{init}}}(\mathsf{safe}\,\mathbf{U}(\langle\langle A\rangle\rangle^{\downarrow}(\mathsf{visual}\,\mathbf{U}\,\mathsf{rescue})))$. Intuitively, this reflects the constraint that the robots cannot recharge their batteries after reaching the position where they can perform rescue while in visual contact with the base. Another example is given by the formula $\langle\langle 1,2\rangle\rangle^{\eta_{\text{init}}}\mathbf{F}(\mathsf{rob}\wedge\langle\langle 1\rangle\rangle^{\downarrow}\mathbf{F}\,\mathsf{escape})$ expressing that the coalition $\{1,2\}$ can cooperate to eventually rob a bank and then agent 1 can ensure to escape on its own using only its remaining resources.

Before we show the decidability of prRAL$^{\mathsf{r}}$ over **iRBM**s in the next section, we make the following observation which follows from [Bulling and Farwer, 2010, Theorem 6]:

Observation 2. *Model checking* prRAL$^{\mathsf{r}}$ *over* **RBM***s is undecidable.*

4 Model-checking prRAL$^{\mathsf{r}}$

In this section we introduce a model-checking algorithm for prRAL$^{\mathsf{r}}$ and prove its correctness. In what follows, we assume that the function $atl(\phi)$ that returns the formula where each $\langle\langle A\rangle\rangle^{\downarrow}$ and $\langle\langle A\rangle\rangle^{\zeta}$ in ϕ is replaced by $\langle\langle A\rangle\rangle$.

Theorem 2. *The model-checking problem for* prRAL$^{\mathsf{r}}$ *over* **iRBM***s is decidable.*

To prove decidability, we give an algorithm which requires as input $\mathfrak{M}$, q, η, ϕ and returns true or false. We prove termination and correctness of the algorithm in Lemmas 1 and 2 below. Given ϕ, we produce a set of subformulas of ϕ, $Sub(\phi)$, in the usual way, except that $\langle\langle A\rangle\rangle^{\downarrow}$ and $\langle\langle A\rangle\rangle^{\zeta}$ modalities are replaced by standard ATL modalities $\langle\langle A\rangle\rangle$. $Sub(\phi)$ is ordered in increasing order of complexity. Note that if a state s is not annotated with the standard ATL modality $\langle\langle A\rangle\rangle$, then it cannot satisfy $\langle\langle A\rangle\rangle^{\downarrow}$ or $\langle\langle A\rangle\rangle^{\zeta}$. Algorithm 1 simply labels the subformulas of ϕ using the standard ATL labelling algorithm [Alur *et al.*, 2002]. It then calls the function STRATEGY to label states with ϕ. (Note that we do not label states with subformulas of ϕ involving $\langle\langle A\rangle\rangle^{\downarrow}$ or $\langle\langle A\rangle\rangle^{\zeta}$ modalities as in [Alechina *et al.*, 2014].)

Algorithm 1 Labelling ϕ

```
procedure LABEL($\mathfrak{M}, \phi, \eta$)
    for $\phi' \in Sub(\phi)$ do
        $[\phi']_{\mathfrak{M}} \leftarrow \{\, q \mid q \in Q \wedge$ ATL-LABEL$(\mathfrak{M}, \phi')\}$
    $[\phi]_{\mathfrak{M}} \leftarrow \{\, q \mid q \in Q \wedge$ STRATEGY$(node_0(q,\eta), \phi)\}$
```

STRATEGY proceeds by depth-first and-or search, processing each coalition modality in turn starting from the outermost modality. Each temporal operator is handled by a separate function: X-STRATEGY for $\mathbf{X}\psi$, U-STRATEGY for $\psi_1\mathbf{U}\psi_2$, and G-STRATEGY for $\mathbf{G}\psi$. We record information about the state of the search in a search tree of nodes. A *node* is a structure which consists of a state of $\mathfrak{M}$, the resources available to the agents A in that state (if any), and a finite path of nodes leading to this node from the root node. Edges in the tree correspond to joint actions by all agents. Note that the resources available to the agents in a state on a path constrain the edges from the corresponding node to be those actions α where for all proponent agents a, $\mathsf{cons}(\alpha)$ is less than or equal to the available resources of agent a. We compare vectors of resources in a usual way, for example $\zeta_a \geq \mathsf{cons}(\alpha_a)$ stands for $\zeta_a(r) \geq \mathsf{cons}(\alpha_a, r)$ for all resources r. For an action tuple σ by $A \subseteq \mathbb{A}\mathrm{gt}$, we write $\mathsf{cons}(\sigma)$ to refer to the tuple $(\mathsf{cons}(\sigma_a))_{a\in A}$. For each node n in the tree, we have a function $s(n)$ which returns its state, $p(n)$ which returns the nodes on the path and $e(n)$ which returns the resource availability for all agents as a result of following $p(n)$. The function $node_0(s,\eta)$ returns the root node, i.e., a node n_0 such that $s(n_0) = s$, $p(n_0) = [\,]$ and $e(n_0) = \eta$. The function $node(n, s', \alpha, A, *)$ where $A \subseteq \mathbb{A}\mathrm{gt}$ is the current set of proponents and $* \in \{\downarrow, \zeta\}$ returns a node n' where $s(n') = s'$, $p(n') = [p(n)\cdot n]$ and

$$e_a(n') = \begin{cases} \zeta_a & \text{if } * = \zeta \\ e_a(n) & \text{if } * = \downarrow \text{ and } a \notin A \\ e_a(n) + \mathsf{prod}(\alpha) - \mathsf{cons}(\alpha) & \text{if } * = \downarrow \text{ and } a \in A \end{cases}$$

X-STRATEGY for $\langle\langle A\rangle\rangle^{\downarrow}\mathbf{X}\psi$ and $\langle\langle A\rangle\rangle^{\zeta}\mathbf{X}\psi$ formulas is shown in Algorithm 3 and is straightforward. After checking if the search should be terminated with false, we simply check if there is an action by A that is possible given current endowment, and where in all outcome states A has a strategy for ψ. G-STRATEGY for $\langle\langle A\rangle\rangle^{\downarrow}\mathbf{G}\phi$ and $\langle\langle A\rangle\rangle^{\zeta}\mathbf{G}\phi$ formulas is shown

Algorithm 2 Strategy

```
function STRATEGY($n, \phi$)
    case $\phi$ is propositional
        return $s(n) \models \phi$
    case $\phi = \psi_1 \wedge \psi_2$
        return STRATEGY$(node_0(s(n), e(n)), \psi_1) \wedge$
                STRATEGY$(node_0(s(n), e(n)), \psi_2)$
    case $\phi = \psi_1 \vee \psi_2$
        return STRATEGY$(node_0(s(n), e(n)), \psi_1) \vee$
                STRATEGY$(node_0(s(n), e(n)), \psi_2)$
    case $\phi = \langle\langle A\rangle\rangle^{\downarrow}\mathbf{X}\psi$
        return X-STRATEGY$(node_0(s(n), e(n)), \phi)$
    case $\phi = \langle\langle A\rangle\rangle^{\zeta}\mathbf{X}\psi$
        return X-STRATEGY$(node_0(s(n), \zeta), \phi)$
    case $\phi = \langle\langle A\rangle\rangle^{\downarrow}\psi_1\mathbf{U}\psi_2$
        return U-STRATEGY$(node_0(s(n), e(n)), \phi)$
    case $\phi = \langle\langle A\rangle\rangle^{\zeta}\psi_1\mathbf{U}\psi_2$
        return U-STRATEGY$(node_0(s(n), \zeta), \phi)$
    case $\phi = \langle\langle A\rangle\rangle^{\downarrow}\mathbf{G}\phi$
        return G-STRATEGY$(node_0(s(n), e(n)), \phi)$
    case $\phi = \langle\langle A\rangle\rangle^{\zeta}\mathbf{G}\phi$
        return G-STRATEGY$(node_0(s(n), \zeta), \phi)$
```

Algorithm 3 X strategy (both types of modalities)

```
function X-STRATEGY($n, \langle\langle A\rangle\rangle^* \mathbf{X}\psi$)
    if $s(n) \not\models atl(\langle\langle A\rangle\rangle^* \mathbf{X}\psi)$ then
        return $false$
    $ActA \leftarrow \{\sigma \in d_A(s(n)) \mid \mathsf{cons}(\sigma) \leq e_A(n)\}$
    for $\sigma \in ActA$ do
        $Act = \{\alpha \in d(s(n)) \mid \alpha_A = \sigma\}$
        $strat \leftarrow true$
        for $\alpha \in Act$ do
            $s' \leftarrow out(s(n), \alpha)$
            $strat \leftarrow strat \wedge$
                STRATEGY($node(n, s', \alpha, A, *), \psi$)
        if $strat$ then
            return $true$
    return $false$
```

in Algorithm 5. Again we check if the search should be terminated with false, either because the standard ATL modality doesn't hold, or because the current path terminates in a resource consuming cycle, or if the current endowment results in ϕ being false. We then check the path for productive loops, and update the endowment if we find one. arb denotes an arbitrary finite value that can be decremented, i.e., where $arb - k < arb$ (unlike for infinity). $e(n)(a, r) = arb$ indicates that the path $p(n)$ terminates in a 'productive loop', which can be traversed multiple times to generate an arbitrary amount of resource r for agent a.[2] If the current path terminates in a nondecreasing loop, we return true. Otherwise we continue the search for a nondecreasing loop. U-STRATEGY for $\langle\langle A\rangle\rangle^{\downarrow}\psi_1\mathbf{U}\psi_2$ and $\langle\langle A\rangle\rangle^{\zeta}\psi_1\mathbf{U}\psi_2$ formulas is shown in Algorithm 4, and is similar to G-STRATEGY. First U-STRATEGY checks whether false should be returned because the ATL version of the formula is false, or the current path has an unproductive loop. We then check the path for productive loops, and update the endowment if we find one. If the ATL version of ψ_2 is true, we try to find a strategy to enforce ψ_2), and if we are successful U-STRATEGY returns true. Otherwise the search continues, because the node where STRATEGY(n, ψ_2) returns true may be found later on the path. Note that if all resources are updated to arb and a repeated call to STRATEGY(n, ψ_2) returns false, the algorithm will return false because of the first check for unproductive loops.

Lemma 1. *Algorithm 2 terminates.*

Proof. (sketch). The proof is by induction on the length of the formula. Calls for propositional formulas clearly terminate. For the inductive step, we need to show that a call for any connective terminates provided calls for lower complexity formulas terminate. Conjunction and disjunction are obvious. X-STRATEGY makes a recursive call to a STRATEGY of a smaller complexity formula after one step. For U-STRATEGY and G-STRATEGY, the recursion calls are infinite iff endowments (considered as a vector of size $|\mathbb{A}\text{gt}| \times |Res|$) are never comparable. However, it can be shown that they are always

[2]In what follows, we denote by $\overline{arb}_A : A \times Res \to \{\infty, arb\}$ an arb or infinite endowment for agents in $A \subseteq \mathbb{A}\text{gt}$.

Algorithm 4 U strategy

```
function U-STRATEGY($n, \langle\langle A\rangle\rangle^* \psi_1\mathbf{U}\psi_2$)
    if $s(n) \not\models atl(\langle\langle A\rangle\rangle^* \psi_1\mathbf{U}\psi_2)$ then
        return $false$
    if $\exists n' \in p(n) : s(n') = s(n) \wedge e_A(n') \geq e_A(n))$ then
        return $false$
    for $(a, r) \in \{a \in A, r \in Res \mid \exists n' \in p(n) :$
                $s(n') = s(n) \wedge e_A(n') \leq e_A(n) \wedge$
                $e(n')(a, r) < e(n)(a, r)\}$ do
        $e(n)(a, r) \leftarrow arb$
    if $s(n) \models atl(\psi_2)$ then
        $strat \leftarrow$ STRATEGY$(n, \psi_2)$
    if $strat$ then
        return $true$
    else
        $ActA \leftarrow \{\sigma \in d_A(s(n)) \mid \mathsf{cons}(\sigma) \leq e_A(n)\}$
        for $\sigma \in ActA$ do
            $Act = \{\alpha \in d(s(n)) \mid \alpha_A = \sigma\}$
            $strat \leftarrow true$
            for $\alpha \in Act$ do
                $s' \leftarrow out(s(n), \alpha)$
                $strat \leftarrow strat \wedge$
                    U-STRATEGY$(node(n, s', \alpha, A, *),$
                                $\langle\langle A\rangle\rangle^* \psi_1\mathbf{U}\psi_2)$
            if $strat$ then
                return $true$
        return $false$
```

Algorithm 5 G strategy

```
function G-STRATEGY($n, \langle\langle A\rangle\rangle^* \mathbf{G}\psi$)
    if $s(n) \not\models \langle\langle A\rangle\rangle^* \mathbf{G}\psi$ then
        return $false$
    if $\exists n' \in p(n) : s(n') = s(n) \wedge (\forall r : e_r(n') \geq e_r(n)) \wedge$
      $(\exists j : e_j(n') > e_j(n))$ or $\neg$STRATEGY$(n, \psi)$ then
        return $false$
    for $(a, r) \in \{a \in A, r \in Res \mid \exists n' \in p(n) :$
                $s(n') = s(n) \wedge e_A(n') \leq e_A(n) \wedge$
                $e(n')(a, r) < e(n)(a, r)\}$ do
        $e(n)(a, r) \leftarrow arb$
    if $\exists n' \in p(n) : s(n') = s(n) \wedge e_A(n') \leq e_A(n))$ then
        return $true$
    $ActA \leftarrow \{\sigma \in d_A(s(n)) \mid \mathsf{cons}(\sigma) \leq e_A(n)\}$
    for $\sigma \in ActA$ do
        $Act = \{\alpha \in d(s(n)) \mid \alpha_A = \sigma\}$
        $strat \leftarrow true$
        for $\alpha \in Act$ do
            $s' \leftarrow out(s(n), \alpha)$
            $strat \leftarrow strat \wedge$
             G-STRATEGY$(node(n, s', \alpha, A, *), \langle\langle A\rangle\rangle^* \mathbf{G}\psi)$
        if $strat$ then
            return $true$
    return $false$
```

going to become comparable after finitely many steps, using the technique in [Alechina *et al.*, 2014] or [Reisig, 1985, p.70]. □

Lemma 2. *Algorithm 2 is correct, that is,* STRATEGY(n,ϕ) *returns true iff* $s(n), e(n) \models \phi$.

Proof. (sketch). The proof is by induction on the modal depth of the formula. The base case (propositional formulas) is immediate. For the inductive step, we assume that the lemma holds for modal depth k, and show it for modal depth $k+1$. The proof of the inductive step is by cases on the main connective of ϕ. The most difficult case is when $\phi = \langle\langle A \rangle\rangle^{\downarrow}\psi' \mathbf{U} \psi$. We need to show that U-STRATEGY$(node_0(s(n), e(n)), \phi)$ returns true iff iff $s(n), e(n) \models \langle\langle A \rangle\rangle^{\downarrow}\psi' \mathbf{U} \psi$.

$(\Rightarrow)$: Let T denote the search tree rooted at $n_0 = node_0(s, \eta)$ when U-STRATEGY$(node_0(q, \eta), \phi)$ returns true. Each node n in T is additionally annotated with the actual [3] endowment $e'(n) : \mathbb{A}\mathrm{gt} \times Res \rightarrow \mathbb{Z} \cup \{\infty\}$ where $(e'(n_0))_i = \eta_i + \sum_{n' \in p(n)} t(a_i(n'))$. Let s_T denote the strategy for A where for each node $n \in T$ with $p(n) = n_1 \ldots n$ $s_T(s(n_1) \ldots s(n)) = (a(n))_A$ for all $n \in T$ and $n_1 \ldots n = p(n)$. Note that, $(e'(n))_i(r)$ can be negative for r and i which means s_T is not a valid strategy. Then, there must be nodes $n', n'' \in p(n)$ such that $(e(n'))_r(i) = arb$ because of endowment in n''. Let $n' = stop_r(n)$ and $n'' = start_r(n)$. Let $T(n)$ denote the subtree of T starting from n. For a resource r, if there is a node n with $(e'(n))_i(r) < 0$ for some i we extend T such that $(e'(n))_i(r) \geq 0$ by repeating the branch between $start_r(n)$ and $end_r(n)$ finitely many times.

Case 1: n is the only node in $T(start_r(n))$ with $(e'(n))_r(a) < 0$ as depicted by Figure 3a.

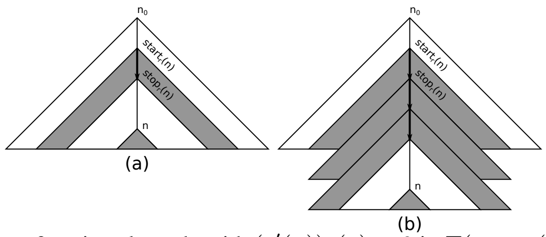


Figure 3: n is only node with $(e'(n))_r(a) < 0$ in $T(start_r(n))$

Assume that the path from $start_r(n)$ to $end_r(n)$ increases r by g. Then, it is necessary to repeat this path $\lceil \frac{|(e'(n))_r(a)|}{g} \rceil$ times, as depicted in Figure 3b.

Case 2: n is not the only node in $T(start_r(n))$ with $(e'(n))_r(a) < 0$. Other nodes n' are either in the subtree $T(stop_r(n))$ (as depicted by Figures 4a) or in the subtree of $T(start_r(n))$ but not $T(stop_r(n))$ (as depicted by Figures 4b). Without loss of generality, we assume that $start_r(n)$ is the ancestor of $start_r(n')$ for any of such n'.

Again, assume that the path from $start_r(n)$ to $end_r(n)$ increases r by g. Then, it is necessary to repeat this path $k = \lceil \frac{|(e'(n))_r(a)|}{g} \rceil$ times, as depicted in Figure 5a. Note that this repetition will also repeat nodes n' which are in the subtree of

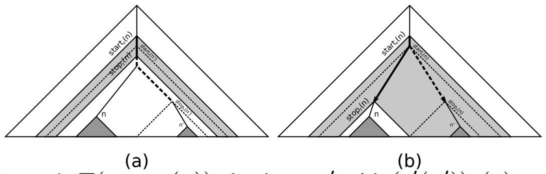


Figure 4: $T(start_r(n))$ also have n' with $(e'(n'))_r(a) < 0$.

$T(start_r(n))$ but not $T(stop_r(n))$ and have $(e'(n'))_r(a) < 0$ as $n'_1, \ldots, n'_k$ depicted in Figure 5b.

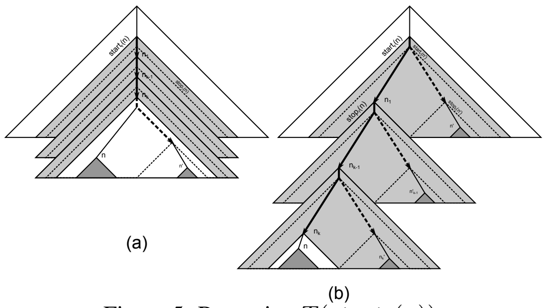


Figure 5: Repeating $T(start_r(n))$.

Let T_1 be the obtained tree. Then, the number of nodes n'' in $T_1(n_{k-1})$ with $(e'(n''))_i(r) < 0$ is strictly less than that in $T(start_r(n))$. Therefore, we can reapply the above construction to obtain a tree T_2 where all nodes n'' in $T_2(n_{k-1})$ have $(e'(n''))_i(r) \geq 0$. These include node n' as depicted in Figure 5a and n'_k as depicted in Figure 5b. Then, we further apply step by step the above construction for nodes $n'_{k-1}, \ldots, n'_1$ and n' in Figure 5b. Finally, we obtain a tree T_3 where all nodes n'' have $(e'(n''))_i(r) \geq 0$. This construction can be repeated for other resources $r' \neq r$ and agents $i' \neq i$. Finally, we obtain a tree T_4 where for all nodes n in T_4, $(e'(n))_i(r) \geq 0$ for all r and i.

$(\Leftarrow)$: As $q, \eta \models \langle\langle A \rangle\rangle^{\downarrow}\psi' \mathbf{U} \psi$, there exists a strategy s_A such that for all $\lambda \in out(q, \eta, s_A, A) : \exists 0 \leq i_\lambda < |\lambda| : \lambda|_Q[i_\lambda] \models \psi$ and $\forall 0 \leq j < i_\lambda : \lambda|_Q[j], \lambda|_{\mathsf{En}}[j] \models \psi'$. Let $T = (V, E)$ be the tree induced by all runs $\lambda[0, i_\lambda]$ for $\lambda \in out(q, \eta, s_A, A)$, i.e., $V = \{\lambda[0, i] \mid \lambda \in out(q, \eta, s_A, A), i \leq i_\lambda\}$ and $E = \{(\lambda[0, i], \lambda'[0, i+1]) \mid \lambda, \lambda' \in out(s, \eta, s_A, A), \lambda[0, i] = \lambda'[0, i], i < i_\lambda\}$. We shall cut T into a search tree which shows that U-STRATEGY$(node_0(q, \eta), \phi)$ returns true. Note that T must be finite and each edge in E corresponds to a join action of all agents.

Initially, let $T_0 = T$, then T_{l+1} is constructed from T_l as follows.

- If there is a node $\lambda[0, i]$ in T_l such that $\exists 0 \leq j < k \leq i : \lambda|_Q[j] = \lambda|_Q[k] \wedge \lambda|_{\mathsf{En}}[j] \geq \lambda|_{\mathsf{En}}[k]$, then T_{l+1} is constructed from T_l by replacing the subtree $T_l[\lambda[j]]$ by $T_l[\lambda[k]]$.
- If there is a node $\lambda[0, i]$ such that $(\lambda|_{\mathsf{En}}[i])_A = \overline{arb}_A$, then T_{l+1} is constructed from T_l by replacing the subtree $T_l[\lambda[i]]$ by the only node $\lambda[i]$.
- If there is a node $\lambda[0, i]$ in T_l such that $\exists 0 \leq j < k \leq i : \lambda|_Q[j] = \lambda|_Q[k] \wedge \lambda|_{\mathsf{En}}[j] \leq \lambda|_{\mathsf{En}}[k] \wedge \lambda|_{\mathsf{En}}[j] \neq \lambda|_{\mathsf{En}}[k]$,

[3] i.e., not containing arb.

then T_{l+1} is constructed from T_l by replacing the endowment η' of all nodes (q', η'') in the subtree $T_l[\lambda[k]]$ by η'' where

$$\eta''_a(r) = \begin{cases} \eta'_a(r) & \text{if } (\lambda|_{\mathsf{En}}[j])_a(r) = \lambda|_{\mathsf{En}}[k]_a(r) \\ arb & \text{if } (\lambda|_{\mathsf{En}}[j])_a(r) < \lambda|_{\mathsf{En}}[k]_a(r) \end{cases}$$

- Otherwise, $T_{l+1} = T_l$, i.e., no more change.

The construction stops when $T_{l+1} = T_l$. Let the resulting tree $T_{l+1} = T'$. For each node $\lambda[0,i]$ in T', we define a node $n_{\lambda[0,i]}$ where: $s(n_{\lambda[0,i]}) = \lambda|_Q[i]$, $p(n_{\lambda[0,i]}) = \lambda|_Q[0,i-1]$ and $e(n_{\lambda[0,i]}) = \lambda|_{\mathsf{En}}[i]$. In the following, we show by induction on the length of $T'(\lambda[0,i])$ that U-STRATEGY$(n_{\lambda[0,i]}, \phi)$ returns true.

Base case: Assume that $\lambda[0,i]$ is a leaf of T', then it is either a leaf from T or an internal node from T that has an $\overline{arb}_A$ endowment for agents in A by the second cut. Then, the condition of the first **if** statement in U-STRATEGY$(n_{\lambda[0,i]}, \phi)$ is false, since $\lambda|_Q[i] \models \psi$ in the former case and $\lambda|_Q[i] \models \langle\!\langle A\rangle\!\rangle\psi'\mathbf{U}\psi$ in the latter case. The condition in the second **if** statement is also false since otherwise T' can be cut further. Similarly, the conditions of the third and fourth **if** statements are true; therefore, U-STRATEGY$(n_{\lambda[0,i]}, \phi)$ returns true.

Induction step: Assume that $\lambda[0,i]$ is not a leaf of T'. Then the condition of the first **if** statement is false, since $q \models \langle\!\langle A\rangle\!\rangle\psi'\mathbf{U}\psi$. The condition of the second **if** statement is also false, since otherwise T' can be cut further. The condition of the third or fourth **if** statement is false since otherwise $\lambda[0,i]$ must have been a leaf of T'. Therefore, the algorithm must enter the second **for** loop. For $\alpha = s_A(\lambda|_Q[0,i]) \in D_A(\lambda[i])$ we have that, for every $s' \in out(\lambda|_Q[i], \alpha)$ with $n' = node(n_{\lambda[0,i]}, s', \alpha, A, \downarrow)$, there must be $\lambda'[0,i+1]$ in T' such that $n' = n_{\lambda'[0,i+1]}$. By the induction hypothesis, U-STRATEGY$(n_{\lambda'[0,i+1]}, \phi)$ returns true. Thus, U-STRATEGY$(n_{\lambda[0,i]}, \phi)$ also returns true.

Obviously, U-STRATEGY$(node_0(q,\eta), \phi)$ returns true since $n_{\lambda[0]} = node_0(q,\eta)$ for any $\lambda[0,i]$ in T'.

The above proof can be adapted to the case $\phi = \langle\!\langle A\rangle\!\rangle^{\zeta}\psi'\mathbf{U}\psi$ by exchanging the role of η and ζ. Finally, for the case $\phi = \langle\!\langle A\rangle\!\rangle^{*}\mathbf{G}\psi$, we can also apply the above proof strategy. In particular, if G-STRATEGY$(node_0(q,\eta), \phi)$ returns true, we construct a strategy from the search tree T where each leaf n of T determines a subtree $T(n')$ where $s(n') = s(n)$ and $e_A(n') \le e_A(n)$ by the third **if** statement. Then, we can replace all leaves n with $T(n')$ repeatedly and end up with an infinite tree T' which induces a strategy $s_{T'}$ to satisfy ϕ. Conversely, if $q, \eta \models \phi$, then ϕ is satisfied by some strategy s_A, and we can cut the tree of computations by s_A into a search tree which shows that G-STRATEGY$(node_0(q,\eta), \phi)$ returns true. □

5 Conclusion

In this paper we investigated the boundary of (un)decidable of logics for verifying resource bounded systems. We proved two undecidability results and identified a significant fragment of Resource Agent Logic with a decidable model checking property. From these results it also follows that the rather natural property on models — that agents can always decide to do nothing — can make model checking decidable.

References

[Alechina *et al.*, 2010] N. Alechina, B. Logan, H. N. Nguyen, and A. Rakib. Resource-bounded alternating-time temporal logic. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 481–488. IFAAMAS, 2010.

[Alechina *et al.*, 2014] N. Alechina, B. Logan, H. N. Nguyen, and F. Raimondi. Decidable model-checking for a resource logic with production of resources. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-2014)*, pages 9–14. ECCAI, IOS Press, 2014.

[Alechina *et al.*, 2015] N. Alechina, B. Logan, H.N. Nguyen, and F. Raimondi. Model-checking for Resource-Bounded ATL with production and consumption of resources. Technical report, ArXiv e-prints 1504.06766, 2015.

[Alur *et al.*, 2002] R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[Bulling and Farwer, 2010] N. Bulling and B. Farwer. On the (un-)decidability of model checking resource-bounded agents. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215, pages 567–572. IOS Press, 2010.

[Bulling and Goranko, 2013] N. Bulling and V. Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *Proceedings 1st International Workshop on Strategic Reasoning, SR 2013*, volume 112 of *EPTCS*, pages 33–41, 2013.

[Della Monica *et al.*, 2013] D. Della Monica, M. Napoli, and M. Parente. Model checking coalitional games in shortage resource scenarios. In *Proceedings of the 4th International Symposium on Games, Automata, Logics and Formal Verification (GandALF 2013*, volume 119 of *EPTCS*, pages 240–255, 2013.

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

[Peterson, 1981] J. L. Peterson. *Petri net theory and the modeling of systems*, volume 132. Prentice-hall Englewood Cliffs (NJ), 1981.

[Reisig, 1985] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985.