Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the complexity of resource-bounded logics

N. Alechina [a], N. Bulling [b], S. Demri [c,*], B. Logan [a]

[a] *University of Nottingham, United Kingdom*
[b] *TU Clausthal, Germany*
[c] *LSV, CNRS, ENS Paris-Saclay, University Paris-Saclay, France*

ARTICLE INFO

Article history:
Received 11 April 2017
Received in revised form 8 January 2018
Accepted 24 January 2018
Available online xxxx

Keywords:
Resource-bounded logics
Vector addition systems with states
Games
Complexity
Temporal logics
Parameters

ABSTRACT

We revisit decidability results for resource-bounded logics and use decision problems on vector addition systems with states (VASS) in order to establish complexity characterisations of (decidable) model checking problems. We show that the model checking problem for the logic RB±ATL is 2EXPTIME-complete by using recent results on alternating VASS (and in EXPTIME when the number of resources is bounded). Moreover, we establish that the model checking problem for RBTL is EXPSPACE-complete. The problem is decidable and of the same complexity for RBTL*, proving a new decidability result as a by-product of the approach. When the number of resources is bounded, the problem is in PSPACE. We also establish that the model checking problem for RB±ATL*, the extension of RB±ATL with arbitrary path formulae, is decidable by a reduction to parity games for single-sided VASS (a variant of alternating VASS). Furthermore, we are able to synthesise values for resource parameters. Hence, the paper establishes formal correspondences between model checking problems for resource-bounded logics advocated in the AI literature and decision problems on alternating VASS, paving the way for more applications and cross-fertilizations.

1. Introduction

Resource-bounded logics. Alternating-time temporal logics such as the logics ATL and ATL* [6] extend the temporal logics CTL and CTL* respectively, by interpreting the formulae on concurrent game structures, a sophisticated extension of labelled transition systems, and by allowing modalities to quantify over strategies for a given coalition of agents. ATL significantly extends CTL but the computational complexity of the model checking problem remains the same. In [6], the labelling algorithm for model checking CTL is extended to ATL, establishing the P-completeness of the model checking problem for ATL [6, Theorem 5.2]. In contrast, the model checking problem for ATL* is 2EXPTIME-complete [6, Theorem 5.6] whereas the problem for CTL* is only PSPACE-complete, see e.g. [23]. The logics ATL and ATL* are well-established formalisms to reason about multi-agent transition systems, and many variants have been proposed over the years, see e.g. [35,5]. We focus here on resource-bounded variants.

Resource-bounded logics [13,12,38,4,3,14] extend alternating-time temporal logics such as ATL [6] by adding transitions that produce and consume resources to the models. As shown in [3], the introduction of implicit counters in the models (i.e., variables interpreted over natural numbers) and the ability to quantify over strategies for a given set of agents can lead to undecidability, or decidability with a very high worst-case upper bound on the complexity of the model checking

* Corresponding author.
E-mail address: demri@lsv.ens-cachan.fr (S. Demri).

https://doi.org/10.1016/j.tcs.2018.01.019
0304-3975/© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

problem. The nature of the strategy modalities means that reasoning about resources has similarities to the analysis of runs of vector addition systems with states (a.k.a. VASS) [34], and more specifically to games on VASS, see e.g. [11]. VASS, and more generally counter machines, are well-known infinite-state systems with many applications in formal verification, see e.g. [10].

Model checking and games on VASS. This paper is an extended version of [2]. In our work, we show how existing results on VASS can be used to analyse the model checking problem for resource-bounded logics. As we recall below, model checking problems on VASS based on temporal logics and games are not always decidable; when they are decidable, they are quite difficult to solve, and complexity characterisations often exist. We briefly recapitulate some of these results below.

Temporal logics on VASS often lead to undecidable model checking problems, see e.g., [24,25], and this is even more true with branching-time temporal logics such as CTL [25], or when the atomic formulae can express properties about counter values [29]. There are exceptions, however. For instance, CTL model checking on one-counter VASS is PSPACE-complete [43, 26] (see also [46]). The control state repeated reachability problem for VASS is shown to be decidable in [30]; this result is generalised to full LTL (for which the atomic formulae correspond exactly to the control states), and the model checking problem for LTL on VASS is shown to be EXPSPACE-complete in [28]. In [30], a strict fragment of LTL restricted to the "infinitely often" temporal operator GF and atomic formulae stating properties on counter values is also shown decidable by a reduction to the reachability problem for VASS.

As far as games for VASS are concerned, the situation is even less encouraging. Indeed, two-player games on VASS in which each player can freely update the counter values are undecidable [11], even with simple winning conditions such as the reachability of a given control state. However, asymmetric VASS games in which at most one player can freely update the counter values and the winning conditions are simple are decidable [40]. For instance, the games on asymmetric VASS with reachability of a control state is shown to be 2EXPTIME-complete in [17], decidable with parity conditions in [1,31] and very recently, a 2EXPTIME upper bound was shown in [16]. The non-termination problem for games on asymmetric VASS is also 2EXPTIME-complete (the upper bound is from [33] and the lower bound is from [17]).

Our motivation. Our main goal in this paper is to establish formal relationships between model checking problems for resource-bounded logics and decision problems for VASS, so that new decidability results can be established for logical problems or new complexity characterisations can be inherited from problems on counter machines. Of course, this is not surprising; resource values and counter values are similar objects, and logics based on concurrent game structures inherently have games in their semantics. Moreover, earlier work has already explored the connections with counter machines, either to obtain undecidability results or to get complexity lower bounds, see e.g. [3]. In this paper, we extend these results to give optimal complexity upper bounds and new decidability results, even for resource-bounded logics with enriched path formulae such as those in CTL* [23] (see also [20]).

Our contributions. As explained above, our approach is to use results from decision problems for alternating VASS (or for its variants with single-sided VASS) to establish new decidability and complexity results for model checking problems on resource-bounded logics. So far, the reductions were rather in the other direction to establish undecidability results (for instance, by reducing the halting problem for Minsky machines).

- The model checking problem for RB±ATL is shown to be 2EXPTIME-complete (see Theorem 2 and Theorem 3). The restriction to a bounded number of resources is also shown to be in EXPTIME. The 2EXPTIME lower bound is obtained by a reduction from the state reachability problem for alternating VASS (AVASS) [17], where the upper bound is shown by a reduction to the state reachability and the non-termination problem for AVASS. We need to consider both target problems in order to reduce our logical problem to questions on AVASS, since the logics can express both reachability and non-termination or invariant properties. So far, the best known result was decidability established in [4] by taking advantage of the well-quasi-ordering $(\mathbb{N}^r, \preceq)$.
- The results for RB±ATL are obtained by using formal relationships between strategies in resource-bounded concurrent game structures and proofs in alternating VASS (the fact that only asymmetric VASS are needed here is the key observation). These relationships are also used to show that the model checking problem for RB±ATL* (a new logic extending RB±ATL as ATL* extends ATL [6]), is decidable, by a reduction to the parity game problem on single-sided VASS [1]. Note that the complexity characterisation of the parity game problem on single-sided VASS was left open in [1,17,33] and it has been recently solved in [16], which allows us to characterise the complexity of the model-checking problem for RB±ATL*. More importantly, we show that resource parameters can be effectively computed in the parameterised version of RB±ATL* thanks to the fact that the Pareto frontier for any parity game on single-sided VASS is computable [1, Theorem 4]. To the best of our knowledge, this is the first time that resource values have been synthesised in resource-bounded logics (see also [32]), and this is done for the rich new logic RB±ATL*.
- The model checking problem for RBTL [12] is shown to be EXPSPACE-complete. The restriction to a bounded number of resources is also shown to be in PSPACE. The model checking problem for RBTL* is shown to be decidable (a new result), and also EXPSPACE-complete (see Theorem 5).

In addition, we also provide complexity characterisations for various fragments of the logic that, e.g., bound the number the resources or the number of agents. For example, the model checking problem for RB±ATL restricted to a single agent is shown to be EXPSPACE-complete (Theorem 4).

2. Logical preliminaries

We write $\mathbb{N}$ (resp. $\mathbb{Z}$) for the set of natural numbers (including 0) (resp. integers) and $[m, m']$ with $m, m' \in \mathbb{Z}$ to denote the set $\{j \in \mathbb{Z} : m \leq j \leq m'\}$. Given a dimension $r \geq 1$ and $a \in \mathbb{Z}$, we write $\vec{a} \in \mathbb{Z}^r$ to denote the vector with all values equal to a. For each $\vec{x} \in \mathbb{Z}^r$, we write $\vec{x}(1), \ldots, \vec{x}(r)$ for the entries of $\vec{x}$. For all $\vec{x}, \vec{y} \in \mathbb{Z}^r$, $\vec{x} \preceq \vec{y} \overset{\text{def}}{\Leftrightarrow}$ for every $i \in [1, r]$, we have $\vec{x}(i) \leq \vec{y}(i)$. We also write $\vec{x} \prec \vec{y}$ when $\vec{x} \preceq \vec{y}$ and $\vec{x} \neq \vec{y}$.

2.1. The logic RB±ATL and its variants

We consider the logics RB±ATL and RB±ATL*. The logic RB±ATL was introduced in [4,5], and extends ATL [6] with resources. RB±ATL* extends RB±ATL to allow path formulae to be any LTL-like formula (see Section 6 for a complete formal definition).

Let PROP be a countably infinite set of atomic propositions. The models for the logics RB±ATL and RB±ATL* are the structures introduced in Definition 1 below. These are concurrent game structures for the logics ATL or ATL* (see e.g. [6]) but enriched with a cost function that specifies how resources are produced or consumed. Intuitively, a concurrent game structure is equipped with r counters and state transitions update their values with increments or decrements.

Definition 1. A *resource-bounded concurrent game structure* $\mathfrak{M}$ is a tuple

$$\mathfrak{M} = (Agt, S, Act, r, \texttt{act}, \texttt{cost}, \delta, Lab)$$

such that:

- Agt is a non-empty finite set of *agents* (by default $Agt = [1, k]$ for some $k \geq 1$);
- S is a non-empty set of states;
- Act is a non-empty set of actions with a distinguished action $\texttt{idle}$;
- $r \geq 1$ is the number of resources;
- $\texttt{act} : S \times Agt \to \mathcal{P}(Act) \setminus \{\emptyset\}$ is an *action manager function*, such that for all s and a, $\texttt{act}(s, a)$ is non-empty and furthermore we have $\texttt{idle} \in \texttt{act}(s, a)$ (some variants give up the existence of $\texttt{idle}$ while obeying the non-emptiness of $\texttt{act}(s, a)$);
- $\texttt{cost} : S \times Agt \times Act \to \mathbb{Z}^r$ is a (partial) cost function; that is, $\texttt{cost}(s, a, \mathtt{a})$ is defined only when $\mathtt{a} \in \texttt{act}(s, a)$,[1] and moreover, we stipulate $\texttt{cost}(s, a, \texttt{idle}) = \vec{0}$;
- $\delta : S \times (Agt \to Act) \to S$ is a (partial) transition function such that δ is defined for a state s and a map $\mathfrak{f} : Agt \to Act$ whenever for all agents $a \in Agt$, we have $\mathfrak{f}(a) \in \texttt{act}(s, a)$;
- $Lab : \text{PROP} \to \mathcal{P}(S)$ is a labelling (the definition can be adapted when finite subsets of PROP are involved).

The map δ is also viewed as a deterministic transition relation with transitions of the form $s \xrightarrow{(\mathtt{a}_1, \ldots, \mathtt{a}_k)} s'$ where $\delta(s, \mathfrak{f}) = s'$ and for all $i \in [1, k] = Agt$, we have $\mathfrak{f}(i) = \mathtt{a}_i$. We say that $\mathfrak{M}$ is *finite* whenever S and Act are finite sets and Lab is restricted to a finite subset of PROP. The *size* of a finite $\mathfrak{M}$ is understood as the size of its encoding when integers are encoded in binary and the maps and sets are encoded in extension, i.e. without any succinct encoding. For example, the size of the part of $\mathfrak{M}$ dedicated to the transition function δ is polynomial in $O(\text{card}(S) \times \text{card}(Act)^{\text{card}(Agt)} \times \text{card}(S))$. Here, we use the standard encoding as in [6] and we do not consider compact encodings in this paper. Fig. 1 illustrates a finite concurrent game structure (costs are omitted).

The idle action was introduced in [4,5], where motivations for requiring a distinguished $\vec{0}$-cost action can be found (in Section 6.5 below, we explain why the idle action is not essential for decidability). Given a *coalition* $A \subseteq Agt$ and a state s, a *joint action* by A is a map $\mathfrak{f} : A \to Act$ such that for all agents $a \in A$, we have $\mathfrak{f}(a) \in \texttt{act}(s, a)$. The set of joint actions by A is denoted $D_A(s)$. A joint action by Agt is a special case of a joint action by $A \subseteq Agt$. Given a state s, the set of joint actions by Agt is simply denoted $D(s)$ (instead of $D_{Agt}(s)$) and the map δ is defined only for such joint actions. We write $\mathfrak{f} \sqsubseteq \mathfrak{g}$ whenever $\mathfrak{g}$ is a conservative extension of $\mathfrak{f}$ (for agents a in the domain of $\mathfrak{f}$, $\mathfrak{g}(a) = \mathfrak{f}(a)$, and the domain of $\mathfrak{g}$ contains at least the agents in the domain of $\mathfrak{f}$).

Given a joint action $\mathfrak{f} \in D_A(s)$, we write $\texttt{out}(s, \mathfrak{f})$ to denote the set below:

$$\texttt{out}(s, \mathfrak{f}) \overset{\text{def}}{=} \{s' \in S \mid \text{there is } \mathfrak{g} \in D(s) \text{ such that } \mathfrak{f} \sqsubseteq \mathfrak{g} \text{ and } s' = \delta(s, \mathfrak{g})\}.$$

[1] Unlike in [4], we adopt the convention that positive costs correspond to resource production, and negative costs to resource consumption.

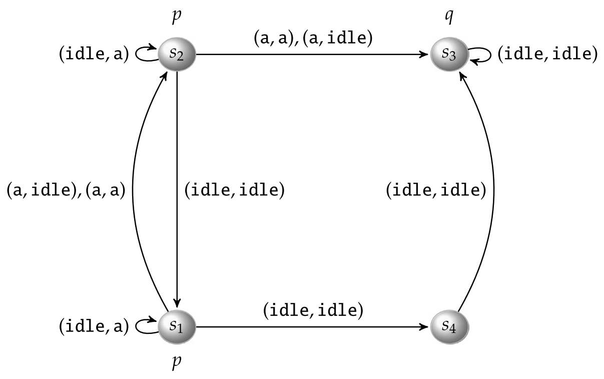

Fig. 1. A concurrent game structure (costs are omitted).

For instance, $\mathtt{out}(s, \mathfrak{f})$ is a singleton set when $\mathfrak{f} \in D(s)$, i.e. if an action is specified for each agent, since δ is a map and not a relation. Given a joint action $\mathfrak{f} \in D_A(s)$ and a state s, the *cost* of a transition from s by $\mathfrak{f}$ (restricted to A by definition) is defined as follows:

$$\mathtt{cost}_A(s, \mathfrak{f}) \overset{\text{def}}{=} \sum_{a \in A} \mathtt{cost}(s, a, \mathfrak{f}(a)).$$

Note that the value $\mathtt{cost}_A(s, \mathfrak{f})$ does not depend on the costs of the actions by the agents in the opponent coalition $(Agt \setminus A)$ (or equivalently, the cost of actions by agents in $(Agt \setminus A)$ is zero). More generally, given $\mathfrak{g} \in D(s)$, we have

$$\mathtt{cost}_A(s, \mathfrak{g}) \overset{\text{def}}{=} \sum_{a \in A} \mathtt{cost}(s, a, \mathfrak{g}(a)).$$

So, the definition of $\mathtt{cost}_A(s, \mathfrak{g})$ can be viewed as a generalisation of the definition of $\mathtt{cost}_A(s, \mathfrak{f})$ with $\mathfrak{f} \in D_A(s)$.

A *computation* λ is a finite sequence or an ω-sequence of the form $s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \ldots$ such that for all $1 \le i+1 < |\lambda|$ we have $s_{i+1} = \delta(s_i, \mathfrak{f}_i)$.[2] Here, $|\lambda|$ denotes the length of λ, each s_i is a state and each $\mathfrak{f}_i$ belongs to $D(s_i)$. For instance, $|s_0 \xrightarrow{\mathfrak{f}_0} s_1 \cdots \xrightarrow{\mathfrak{f}_{n-1}} s_n| = n+1$ and $|s_0 \xrightarrow{\mathfrak{f}_0} s_1 \cdots \xrightarrow{\mathfrak{f}_{n-1}} \cdots| = \omega$ for any infinite computation. A *strategy* F_A for the coalition A is a map from the set of finite computations to the set of joint actions of A such that

$$F_A(s_0 \xrightarrow{\mathfrak{f}_0} s_1 \cdots \xrightarrow{\mathfrak{f}_{n-1}} s_n) \in D_A(s_n).$$

The notion of strategy we define here is somewhat stronger than in ATL, as agents can make their strategy dependent on actions. So, even if the sequence of states is the same, actions assigned by a strategy can be different depending on the sequence of actions. This does not make a difference if purely qualitative ATL formulae are considered.

A computation $\lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \cdots$ respects the strategy F_A iff for all $i < |\lambda|$, we have, $s_{i+1} \in \mathtt{out}(s_i, F_A(s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \xrightarrow{\mathfrak{f}_{i-1}} s_i))$. A computation λ that respects F_A is *maximal* whenever it cannot be extended further while respecting the strategy. Note that maximal computations respecting F_A are infinite. The set of all maximal computations that respect the strategy F_A that start at the state s is denoted by $\mathtt{Comp}(s, F_A)$. So far, no resource value has been involved in computations. Below, we shall quantify over maximal computations that respect a strategy, and therefore for defining a strategy we can restrict ourselves to finite computations that respect it so far.

Given a bound $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ and a computation $\lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \xrightarrow{\mathfrak{f}_1} s_2 \ldots$ in $\mathtt{Comp}(s, F_A)$, let the resource availability at step $i < |\lambda|$ be defined as follows: $\vec{v}_0 \overset{\text{def}}{=} \vec{b}$ and for all $i+1 < |\lambda|$, $\vec{v}_{i+1} \overset{\text{def}}{=} \mathtt{cost}_A(s_i, \mathfrak{f}_i) + \vec{v}_i$ (assuming that $n + \omega = \omega$ for any $n \in \mathbb{Z}$). Then, λ is $\vec{b}$*-consistent* iff for all $i < |\lambda|$, $\vec{v}_i \in (\mathbb{N} \cup \{\omega\})^r$. If $\vec{b}(i) = \omega$, we have an infinite supply of the ith resource and effectively disregard what happens on the ith resource. Since the resource availability of the sequence depends only on the agents in A, this is called the *proponent restriction condition*. This condition is very similar to that found in runs of VASS with the sequence of update vectors $\mathtt{cost}_A(s_0, \mathfrak{f}_0), \mathtt{cost}_A(s_1, \mathfrak{f}_1), \ldots$. Note also that the above condition is slightly different from the one in [5] but equivalent. We have decided to use our notation in order to more easily show the relationships with VASS decision problems.

[2] Each transition between two successive states is labelled by a joint action: this is not strictly necessary for the development below, but it provides a more general notion that might be used in other contexts (for example, if the winning condition of strategies depends on the actions of all the agents and not only on those for the agents in A or on the visited states).

The set of all the $\vec{b}$-consistent (infinite) computations is denoted by $\texttt{Comp}(s, F_A, \vec{b})$. A *$\vec{b}$-strategy F_A with respect to s* is a strategy such that $\texttt{Comp}(s, F_A) = \texttt{Comp}(s, F_A, \vec{b})$. This definition also differs slightly from that given in [5]; the notion of *$\vec{b}$-strategy* in [5] is not relative to a state and therefore the equality should hold for all states.

With the main definitions of resource-bounded concurrent game structures and strategies in hand, we can now present the logic RB±ATL. Given a set of agents $Agt = \{1, \dots, k\}$ and $r \geq 1$, we write RB±ATL(Agt, r) to denote the resource-bounded logic with k agents and r resources whose models are resource-bounded concurrent game structures with the same parameters. Formulae of RB±ATL(Agt, r) are defined according to the grammar below:

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A^{\vec{b}} \rangle\rangle\, \mathsf{X}\phi \mid \langle\langle A^{\vec{b}} \rangle\rangle\, \mathsf{G}\phi \mid \langle\langle A^{\vec{b}} \rangle\rangle\, \phi \mathsf{U} \phi,$$

where $p \in$ PROP, $A \subseteq Agt$ and $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$.

The meaning of $\langle\langle A^{\vec{b}} \rangle\rangle\, \mathsf{X}\phi$ is that A have a strategy which can be executed within the resource bound $\vec{b}$ to enforce ϕ in the next state (essentially, A have a joint action which consumes less than $\vec{b}$ resources and is guaranteed to achieve a ϕ-state whatever the opponents do). $\langle\langle A^{\vec{b}} \rangle\rangle\, \mathsf{G}\phi$ means that A have a strategy which can be executed within the resource bound $\vec{b}$ to maintain ϕ forever. $\langle\langle A^{\vec{b}} \rangle\rangle\, \phi_1 \mathsf{U} \phi_2$ means that A have a strategy which can be executed within the resource bound $\vec{b}$ to reach a ϕ_2-state while maintaining ϕ_1. Since resource bounds may include ω, which means that there is no resource bound on the strategy, RB±ATL includes ATL. Similarly to ATL, the language of RB±ATL includes G rather than the release operator R, even though R is not expressible in ATL [35] (hence also not in RB±ATL). The main reason for the choice of operators is consistency with ATL as defined in [6] and RB±ATL as defined in [4], and the fact that this choice of operators appears to be sufficient to describe properties of interest in verification problems. An example property that can be expressed in RB±ATL is 'Agents a and b have a strategy which requires at most 100 units of energy to reach a position where a can stay in orbit forever without requiring any additional energy, and while they are reaching this position they can also always abort the mission, again with no energy requirement'. This can be expressed as

$$\langle\langle \{a, b\}^{100} \rangle\rangle (\langle\langle \{a, b\}^{0} \rangle\rangle\, \mathsf{X}\, abort)\, \mathsf{U}\, (\langle\langle \{a\}^{0} \rangle\rangle\, \mathsf{G}\, orbit).$$

The size of a formula is computed from a DAG representation and the integers are encoded in binary. Note that forthcoming hardness results do not use the conciseness of the DAG representation (with respect to the tree representation). The satisfaction relation $\models$ is defined inductively as follows assuming that $\mathfrak{M}$ is an RB±ATL(Agt, r) model (we omit the obvious cases for the Boolean connectives):

$$\begin{array}{lcl}
\mathfrak{M}, s \models p & \overset{\text{def}}{\Leftrightarrow} & s \in Lab(p) \\
\mathfrak{M}, s \models \langle\langle A^{\vec{b}} \rangle\rangle \mathsf{X}\phi & \overset{\text{def}}{\Leftrightarrow} & \text{there is a } \vec{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that} \\
 & & \text{for all } s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \texttt{Comp}(s, F_A), \text{ we have } \mathfrak{M}, s_1 \models \phi \\
\mathfrak{M}, s \models \langle\langle A^{\vec{b}} \rangle\rangle \mathsf{G}\phi & \overset{\text{def}}{\Leftrightarrow} & \text{there is a } \vec{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that} \\
 & & \text{for all } \lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \texttt{Comp}(s, F_A), \text{ for all } i < |\lambda|, \\
 & & \text{we have } \mathfrak{M}, s_i \models \phi \\
\mathfrak{M}, s \models \langle\langle A^{\vec{b}} \rangle\rangle \phi_1 \mathsf{U} \phi_2 & \overset{\text{def}}{\Leftrightarrow} & \text{there is a } \vec{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that for all} \\
 & & \lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \texttt{Comp}(s, F_A), \text{ there is some } i < |\lambda| \\
 & & \text{such that } \mathfrak{M}, s_i \models \phi_2 \text{ and} \\
 & & \text{for all } j \in [0, i-1], \text{ we have } \mathfrak{M}, s_j \models \phi_1.
\end{array}$$

Standard semantics for temporal operators. It is worth noting that since all the maximal computations are infinite, the index i involved for clauses related to $\langle\langle A^{\vec{b}} \rangle\rangle \mathsf{G}$ or $\langle\langle A^{\vec{b}} \rangle\rangle \mathsf{U}$ can take any value in $\mathbb{N}$. The temporal operators X, G and U have their standard meaning from linear-time temporal logic LTL. CTL formulae can be expressed by RB±ATL formulae (as it is also classically the case with ATL) if resources are omitted from resource-bounded concurrent game structures thanks to correspondences of the form $\langle\langle Agt^{\vec{\omega}} \rangle\rangle \mathsf{G}p \approx \mathsf{EG}p$, $\langle\langle \emptyset^{\vec{\omega}} \rangle\rangle p_1 \mathsf{U} p_2 \approx \mathsf{A}(p_1 \mathsf{U} p_2)$ and $\langle\langle \emptyset^{\vec{\omega}} \rangle\rangle \mathsf{G}p \approx \mathsf{AG}p$, etc.

Ability to safely extend any finite strategy. The presence of the idle action allows a (partially defined) strategy to be extended to an infinite strategy as soon as a formula is satisfied along the computations. For instance, $\mathfrak{M}, s \models \langle\langle A^{\vec{b}} \rangle\rangle \mathsf{X}\phi$ is equivalent to the existence of $\mathfrak{f} \in D_A(s)$ such that for all $\mathfrak{g} \sqsupseteq \mathfrak{f}$, we have $\mathfrak{M}, s' \models \phi$ with $\delta(s, \mathfrak{g}) = s'$ and $\vec{b} + \texttt{cost}_A(s, \mathfrak{f}) \succeq \vec{0}$.

Upward closure. Observe also that a strategy modality $\langle\langle A^{\vec{b}} \rangle\rangle$ reduces the impact of the function $\texttt{cost}$ in two ways. If the ith component of $\vec{b}$ is equal to ω, then there are no constraints on the ith resource along the computation. Moreover, the restriction of $\texttt{cost}$ to proponent agents in A means that the actions of the opponents cost nothing and are always available. In addition, it is worth noting that $\langle\langle A^{\vec{b}} \rangle\rangle \phi \mathsf{U} \psi \Rightarrow \langle\langle A^{\vec{b}'} \rangle\rangle \phi \mathsf{U} \psi$ and $\langle\langle A^{\vec{b}} \rangle\rangle \mathsf{G}\phi \Rightarrow \langle\langle A^{\vec{b}'} \rangle\rangle \mathsf{G}\phi$ are valid (with $\phi_1 \Rightarrow \psi_2$

an abbreviation for $\neg(\psi_1 \wedge \neg\psi_2)$) whenever $\vec{b} \preceq \vec{b}'$, and therefore whenever $\mathfrak{M}, s \models \langle\langle A^{\vec{b}}\rangle\rangle \phi \mathsf{U} \psi$ there is a finite set of minimal elements $\vec{m} \in (\mathbb{N} \cup \{\omega\})^r$ (with respect to $\preceq$) such that $\mathfrak{M}, s \models \langle\langle A^{\vec{m}}\rangle\rangle \phi \mathsf{U} \psi$, similarly for $\langle\langle A^{\vec{b}}\rangle\rangle \mathsf{G} \phi$ (by Dickson's Lemma [22] every upward closed set of $(\mathbb{N} \cup \{\omega\})^r$ admits a finite basis of minimal elements; see also the notion of *Pareto frontier* in Section 6).

Alternative semantics. In the definition of the satisfaction relation $\models$ for RB±ATL, in the clauses for a strategy modality followed by a temporal formula, there is an existential quantification over a $\vec{b}$-strategy F_A with respect to a state s followed by a universal quantification over all the computations in $\mathtt{Comp}(s, F_A)$. By definition, $\mathtt{Comp}(s, F_A) = \mathtt{Comp}(s, F_A, \vec{b})$, and therefore all the computations involved in the universal quantification are maximal and infinite, and, of course, all the underlying resource availabilities along the computations are non-negative. Alternative definitions have been considered in the literature that separate maximality from infinity, leading sometimes to different decidability results. For instance, with the *infinite semantics*, the existential quantification is over a strategy (that is not necessarily a $\vec{b}$-strategy), and the universal quantification is made only over *infinite* computations that respect the strategy (typically nothing is required on maximal and finite computations). Under certain assumptions, this may lead to undecidability, see e.g. [5, Section 6]. Similarly, with the *finite semantics*, the existential quantification is over a strategy (that is not necessarily a $\vec{b}$-strategy), and the universal quantification is made only over *maximal* (either finite or infinite) computations that respect the strategy. In this paper, we shall not investigate logics with these alternative semantics, as from our technical developments of alternating VASS, we can easily derive new decision problems on alternating VASS that correspond to such logical variants. Another way to define alternative semantics is to change the notion of resource-bounded concurrent game structures; for example, by assuming that there is no distinguished idle action, or requiring that the action manager function is of the form $\mathtt{act} : S \times Agt \to \mathcal{P}(Act)$, i.e. an agent may be unable to choose an action from a given state (because the action manager returns an empty set of actions in that state). In what follows, we shall investigate these variants by simply adapting the techniques for RB±ATL with the standard semantics defined above.

The *model checking problem for RB±ATL* is defined as follows:

Input: $k, r \geq 1$ (in unary), a formula ϕ in RB±ATL($[1,k], r$), a finite RB±ATL($[1,k], r$) model $\mathfrak{M}$ and a state s,
Question: $\mathfrak{M}, s \models \phi$?

The encoding of the values in k and r in unary is not essential here, since, if transitions are represented explicitly, the size of $\mathfrak{M}$ is greater than $k + r$.

Proposition 1. *[4, Theorem 1] The model checking problem for RB±ATL is decidable.*

A key contribution of this paper is characterising the computational complexity of the model checking problem for RB±ATL. Obviously, RB±ATL is a quantitative extension of ATL, and whereas the satisfaction of ATL formulae can be restricted to positional strategies (i.e., actions are chosen based on the current state rather than histories), the satisfaction of RB±ATL formulae may require non-positional strategies in order to keep the amount of each resource above zero.

3. Problems on vector addition systems with states (VASS)

In this section, we recall known complexity/decidability results for model checking and games on VASS, and state necessary complexity characterisations that will be used in the sequel. We then show that using optimal decision procedures for VASS problems as black boxes leads to optimal decision procedures for model checking problems of resource-bounded logics.

3.1. Alternating VASS

A binary tree $\mathfrak{T}$, which may contain nodes with (only) one child, is a non-empty subset of $\{1,2\}^*$ such that, for all $\mathfrak{n} \in \{1,2\}^*$ and $i \in \{1,2\}$, $\mathfrak{n} \cdot i \in \mathfrak{T}$ implies $\mathfrak{n} \in \mathfrak{T}$ and, $\mathfrak{n} \cdot 2 \in \mathfrak{T}$ implies $\mathfrak{n} \cdot 1 \in \mathfrak{T}$. The nodes of $\mathfrak{T}$ are its *elements*. The root of $\mathfrak{T}$ is ε, the empty word. All notions such as parent, first child, second child, subtree and leaf, have their standard meanings. The height of $\mathfrak{T}$ is the length, i.e. the number of nodes, of the longest simple path from the root to a leaf. An *alternating VASS* (AVASS) [17] is a tuple $\mathcal{A} = (Q, r, R_1, R_2)$ such that:

- Q is a non-empty finite set of *locations* (a.k.a. *control states*) and $r \geq 0$ is the number of resource values;
- R_1 is a finite subset of $Q \times \mathbb{Z}^r \times Q$ (*unary rules*);
- R_2 is a (finite) subset of Q^3 (*fork rules*).

A *derivation skeleton* of $\mathcal{A}$ is a labelling $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\perp\})$ such that:

- $\mathfrak{T}$ is a binary tree;
- if $\mathfrak{n}$ has one child in $\mathfrak{T}$, then $\mathcal{D}(\mathfrak{n}) \in R_1$;

- if $\mathfrak{n}$ has two children in $\mathfrak{T}$, then $\mathcal{D}(\mathfrak{n}) \in R_2$;
- if $\mathfrak{n}$ is a leaf in $\mathfrak{T}$, then $\mathcal{D}(\mathfrak{n}) = \perp$;
- if $\mathcal{D}(\mathfrak{n}) = (q, \vec{u}, q')$ and $\mathcal{D}(\mathfrak{n} \cdot 1) \in R_1 \cup R_2$, then the first location of $\mathcal{D}(\mathfrak{n} \cdot 1)$ is q';
- if $\mathcal{D}(\mathfrak{n}) = (q, q_1, q_2)$ and $\mathcal{D}(\mathfrak{n} \cdot i) \in R_1 \cup R_2$ for some $i \in \{1, 2\}$, then the first location of $\mathcal{D}(\mathfrak{n} \cdot i)$ is q_i.

A *derivation* of $\mathcal{A}$ based on $\mathcal{D}$ is a labelling $\hat{\mathcal{D}} : \mathfrak{T} \to Q \times \mathbb{Z}^r$ such that:

- if $\mathfrak{n}$ has one child $\mathfrak{n}'$ in $\mathfrak{T}$, $\mathcal{D}(\mathfrak{n}) = (q, \vec{u}, q')$ and $\hat{\mathcal{D}}(\mathfrak{n}) = (q, \vec{v})$, then $\hat{\mathcal{D}}(\mathfrak{n}') = (q', \vec{u} + \vec{v})$;
- if $\mathfrak{n}$ has two children $\mathfrak{n}'$ and $\mathfrak{n}''$ in $\mathfrak{T}$, $\mathcal{D}(\mathfrak{n}) = (q, q_1, q_2)$ and $\hat{\mathcal{D}}(\mathfrak{n}) = (q, \vec{v})$, then $\hat{\mathcal{D}}(\mathfrak{n}') = (q_1, \vec{v})$ and $\hat{\mathcal{D}}(\mathfrak{n}'') = (q_2, \vec{v})$.

Note that fork rules do not update the resources, and therefore there is an asymmetry between unary rules and fork rules. This will be a very useful feature later, when dealing with the proponent restriction condition in RB±ATL. Unlike branching VASS (see e.g., [45,21]), the fork rules have no effect on the counter values.

A derivation $\hat{\mathcal{D}}$ based on $\mathcal{D}$ is *admissible* whenever $\hat{\mathcal{D}} : \mathfrak{T} \to Q \times \mathbb{N}^r$, i.e., only natural numbers occur in it. An admissible derivation is also called a *proof*. Above, we introduced the primitive notion of computations and their restriction to $\vec{b}$-consistent computations. Similarly, the primitive notion of derivations can be restricted to proofs (a kind of "$\vec{0}$-consistency").

As an illustration, we present a proof from an alternating VASS having at least the unary rules $\mathfrak{r}_1 = q_1 \xrightarrow{(-1,+3)} q_0$ and $\mathfrak{r}_3 = q_2 \xrightarrow{(+3,+3)} q_3$, and the fork rule $\mathfrak{r}_2 = q_0 \to q_1, q_2$.

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ (q_3,(4,8))\end{array}}{(q_2,(1,5))}\,(\mathfrak{r}_3)\qquad\dfrac{\begin{array}{c}\vdots\\ (q_0,(0,8))\end{array}}{(q_1,(1,5))}\,(\mathfrak{r}_1)}{\dfrac{(q_0,(1,5))}{(q_1,(2,2))}\,(\mathfrak{r}_1)}\,(\mathfrak{r}_2)$$

Before presenting the decision problems on AVASS, we state a simple property that will be used in the sequel.

Lemma 1. *Given a derivation skeleton $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\perp\})$ such that $\mathcal{D}(\varepsilon)$ is a rule whose first location is q and $(q, \vec{b}) \in Q \times \mathbb{Z}^r$, there is a unique derivation $\hat{\mathcal{D}}$ of $\mathcal{A}$ based on $\mathcal{D}$ such that $\hat{\mathcal{D}}(\varepsilon) = (q, \vec{b})$.*

Indeed, once the rules are provided by $\mathcal{D}$, the root value $(q, \vec{b})$ determines all the values of the derivation since the way $\hat{\mathcal{D}}(\mathfrak{n})$ is defined remains essentially deterministic. The *state reachability problem* for AVASS is defined as follows:

Input: An alternating VASS $\mathcal{A}$ and control states q_0 and q_f.
Question: Is there a finite proof of $\mathcal{A}$ whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$?

When $\mathcal{A}$ has no fork rules, $\mathcal{A}$ is essentially a VASS [34] and the above problem is an instance of the coverability problem known to be EXPSPACE-complete [37,39] (see also [8,19]). The *non-termination problem* for AVASS is defined as follows:

Input: An alternating VASS $\mathcal{A}$ and a control state q_0.
Question: Is there a proof of $\mathcal{A}$ whose root is equal to $(q_0, \vec{0})$ and all the maximal branches are infinite?

Proposition 2. *[17,33] The state reachability and non-termination problems for AVASS are 2EXPTIME-complete.*

The decidability of these problems was first established in [40] by using monotonicity of the games. The 2EXPTIME upper bound is preserved if we assume that the root is labelled by $(q_0, \vec{b})$ with $\vec{b} \in \mathbb{N}^r$ encoded with a binary representation (see Lemma 7 below).

In the sequel, we shall also admit fork rules of any arity $\beta \geq 1$ and therefore in such slightly extended AVASS, the set of fork rules R_2 is a finite subset of $\bigcup_{\beta \geq 2} Q^{\beta}$. The notions of derivation skeleton, derivation and proof are also changed to refer to general trees $\mathfrak{T} \subseteq (\mathbb{N} \setminus \{0\})^*$. The set of finite words $\mathfrak{T} \subseteq (\mathbb{N} \setminus \{0\})^*$ is a (not necessarily binary) *tree* iff for all $\mathfrak{n} \in (\mathbb{N} \setminus \{0\})^*$ and $i \in (\mathbb{N} \setminus \{0\})$, $\mathfrak{n} \cdot i \in \mathfrak{T}$ implies $\mathfrak{n} \in \mathfrak{T}$, and $\mathfrak{n} \cdot i \in \mathfrak{T}$ and $i > 1$ imply $\mathfrak{n} \cdot (i-1) \in \mathfrak{T}$. In the remainder of the paper, by AVASS we mean such an extended AVASS with fork rules of arbitrary arity.

3.2. Model checking problems

A VASS can be defined as an alternating VASS without any fork rules, and therefore we write it $\mathcal{V} = (Q, r, R)$ where R is a finite set of unary rules. Given a VASS $\mathcal{V}$, its transition system $\mathfrak{TS}(\mathcal{V}) \overset{\text{def}}{=} (\mathfrak{W}, \to, Lab)$ is such that:

- $\mathfrak{W} \overset{\text{def}}{=} Q \times \mathbb{N}^r$;
- *Lab* is a labelling with elements of Q also understood as propositional variables and $Lab(q) \overset{\text{def}}{=} \{q\} \times \mathbb{N}^r$ (the truth value of the atomic formula q on a configuration in $Q \times \mathbb{N}^r$ only depends on the control state);
- $\rightarrow$ is a binary relation on $\mathfrak{W}$ such that $(q, \vec{v}) \rightarrow (q', \vec{z})$ iff there is a unary rule $(q, \vec{u}, q')$ in R such that $\vec{z} = \vec{u} + \vec{v}$ where '$+$' is the component-wise addition on $\mathbb{N}^r$.

As usual, we also write $\xrightarrow{*}$ to denote the reflexive and transitive closure of $\rightarrow$. Since $\mathfrak{TS}(\mathcal{V})$ is a Kripke-style structure, it can be used to interpret modal or temporal formulae where the atomic formulae refer to locations, e.g., formulae of the temporal logics LTL or CTL (see also [18]). Recall that LTL and CTL are both fragments of CTL*. Alternating-time temporal logics ATL or ATL* strictly extend CTL or CTL* respectively. Hence complexity hardness results for temporal logics can be lifted to alternating-time logics. We adopt this approach.

We first recall some results that will be useful in the sequel.

Proposition 3. *The model checking problem for LTL on VASS is* EXPSPACE*-complete (the atomic formulae are control states) and it is in* PSPACE *for a fixed number of resources [28].*

EXPSPACE-hardness of model checking on VASS already follows from EXPSPACE-hardness of the state reachability problem for VASS [37], as state reachability is a subproblem of the model checking problem for VASS (consider the LTL formula $\mathsf{F} q_f$).

4. On the complexity of RB±ATL

In this section, we show how to solve the model checking problem for RB±ATL by solving instances of decision problems for alternating VASS using a labelling algorithm. The size of the AVASS problem instance is linear in the input resource-bounded concurrent game structure, and the number of calls to the algorithms solving instances of decision problems for AVASS is also linear in the size of the input formulae. As far as worst-case complexity bounds are concerned, this is probably the best we can hope for. At a high level, our results relate model checking problems for resource-bounded logics in AI and verification games. Even though this first correspondence, as stated, does not provide a good intuition, at a technical level, this reduces to three key correspondences. First, the proponent restriction condition in RB±ATL corresponds to the fact that, in AVASS, only unary rules can update the counter values. This is crucial to our results, but alone it is not sufficient. Second, each $\vec{b}$-strategy F_A generates a set of computations that can be represented as a finitely branching tree with infinite branches, which corresponds precisely to the proofs in AVASS (see e.g., Theorem 1 below). Third, roughly speaking, temporal formulae in the scope of a strategy modality correspond to acceptance conditions on branches of the proofs (admissible computations) extracted from the AVASS. In the remainder of this section, we develop these correspondences (summarised in the table below) in detail.

RB±ATL	Alternating VASS
Logic in AI	Verification games
proponent restriction condition	updates in R_1 / no update in R_2
computation tree for F_A	proof
formulae in the scope of $\langle\langle A^{\vec{b}} \rangle\rangle$	monotone objectives

4.1. *Structural analysis of strategies and proofs*

We first establish the necessary formal relationships between strategies in resource-bounded concurrent game structures, and proofs in alternating VASS. The technical developments are not conceptually difficult, but they allow us to derive results that are helpful in solving the model checking problem for RB±ATL using decision procedures on AVASS. Our approach also allows us to pose and solve new model checking problems (see e.g. Section 6 and Section 6.6).

Let $\mathfrak{M}$ be a finite resource-bounded concurrent game structure, $A \subseteq Agt$ be a coalition and $s^\star$ be a state. We construct an alternating VASS $\mathcal{A}_{\mathfrak{M},A,s^\star}$ such that the set of computations starting in $s^\star$ and respecting some strategy F_A corresponds precisely to a derivation skeleton whose root is labelled by a unary rule with first state $s^\star$. Moreover, if F_A is a $\vec{b}$-strategy w.r.t. $s^\star$, then the derivation skeleton can be turned into a proof whose root is labelled by $(s^\star, \vec{b})$. This implies that fork rules can have any arity greater than one, and components can have the value ω, where ω is a value that remains constant. Nevertheless, note that below, the construction of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ does not depend on any strategy.

Given $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, Lab)$ and a distinguished state $s^\star \in S$, the AVASS $\mathcal{A}_{\mathfrak{M},A,s^\star} \overset{\text{def}}{=} (Q, r, R_1, R_2)$ is built as follows:

$$Q \overset{\text{def}}{=} \{s^\star\} \cup \{(s', \mathfrak{f}) \mid s' \in S,\ \mathfrak{f} \in D_A(s')\} \cup \{(\mathfrak{g}, s') \mid s', s'' \in S,\ \mathfrak{g} \in D(s''),\ \delta(s'', \mathfrak{g}) = s'\}.$$

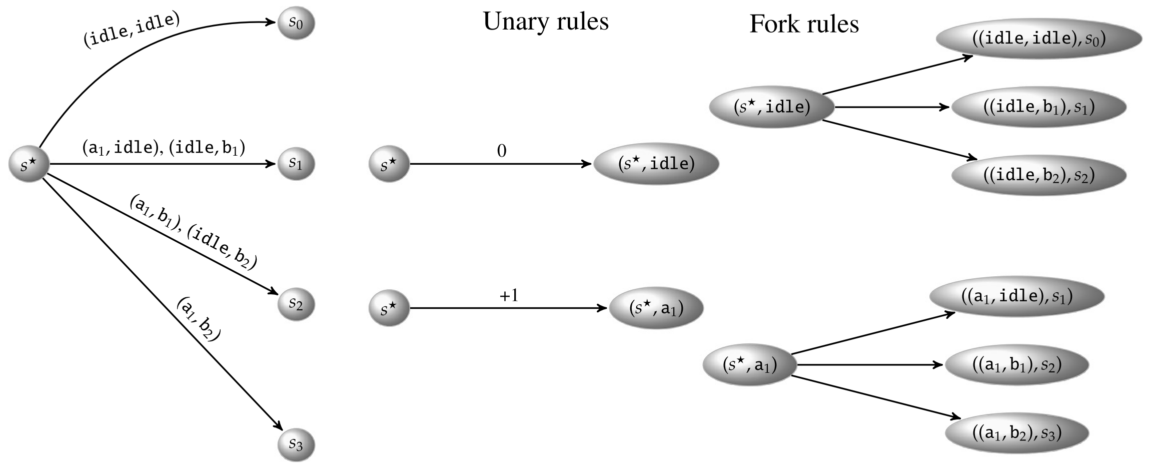

Fig. 2. Transitions and its associated unary and fork rules.

- The set of unary rules R_1 contains the following elements:
  - For all $\mathfrak{f} \in D_A(s^\star)$, $(s^\star, \texttt{cost}_A(s^\star, \mathfrak{f}), (s^\star, \mathfrak{f}))$.
  - For all $(\mathfrak{g}, s') \in Q$, for all $\mathfrak{f} \in D_A(s')$, $((\mathfrak{g}, s'), \texttt{cost}_A(s', \mathfrak{f}), (s', \mathfrak{f}))$.
- The set of fork rules R_2 contains the following elements.
  - For all $(s', \mathfrak{f}) \in Q$, let $\{(\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)\} = \{(\mathfrak{g}, s'') \in S \mid s'' = \delta(s', \mathfrak{g}),\ \mathfrak{g} \in D(s'),\ \mathfrak{f} \sqsubseteq \mathfrak{g}\}$. This set is non-empty because an action manager always returns a non-empty set of actions. We add the α-ary fork rule

$$((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)).$$

In order to define the rule unambiguously, we assume an arbitrary linear ordering on the set Q and on the set of joint actions $\mathfrak{g} : Agt \to Act$.

The following observations shall be useful in the sequel.

- $\mathrm{card}(Q)$ is quadratic in the size of $\mathfrak{M}$, $\mathrm{card}(R_1 \cup R_2)$ is polynomial in the size of $\mathfrak{M}$ and each vector in a rule of R_1 has values at most exponential in the size of $\mathfrak{M}$. Consequently, the size of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ is polynomial in the size of $\mathfrak{M}$.
- $s^\star$ has a special status in Q simply because any proof whose root configuration contains $s^\star$ has no predecessor configuration.
- By construction, any derivation skeleton from $\mathcal{A}_{\mathfrak{M},A,s^\star}$ has to alternate the rules in R_1 and the rules in R_2. This property will be used to slightly simplify developments below.
- For every $(s', \mathfrak{f})$ in Q, there is a unique fork rule starting from $(s', \mathfrak{f})$.
- The construction also applies in degenerated cases, i.e., when $A = Agt$ or when $A = \emptyset$ (assuming that $\texttt{cost}(s', \mathfrak{f}) = \vec{0}$ for the unique $\mathfrak{f} \in D_\emptyset(s')$).

In Fig. 2, we illustrate how transitions from the state $s^\star$ are turned into unary rules and fork rules (in the example, $Agt = \{1, 2\}$, $A = \{1\}$, $\texttt{act}(s^\star, 1) = \{\texttt{idle}, \mathrm{a}_1\}$, $\texttt{act}(s^\star, 2) = \{\texttt{idle}, \mathrm{b}_1, \mathrm{b}_2\}$, and $\texttt{cost}(s^\star, 1, \mathrm{a}_1) = +1$).

Given an infinite computation $\lambda = s_0 \xrightarrow{\mathfrak{g}_1} s_1 \xrightarrow{\mathfrak{g}_2} s_2 \ldots$ starting in $s^\star = s_0$ and respecting F_A, we can associate it with an infinite sequence (which we call an *extended computation*)

$$\texttt{ext}(\lambda, F_A) \overset{\mathrm{def}}{=} s_0 \xrightarrow{\vec{u}_0} (s_0, \mathfrak{f}_0) \to (\mathfrak{g}_1, s_1) \xrightarrow{\vec{u}_1} (s_1, \mathfrak{f}_1) \to (\mathfrak{g}_2, s_2) \xrightarrow{\vec{u}_2} (s_2, \mathfrak{f}_2) \to (\mathfrak{g}_3, s_3) \cdots$$

where $s_0 = s^\star$, and for all $n \geq 0$, $F_A(s_0 \xrightarrow{\mathfrak{g}_1} s_1 \ldots \xrightarrow{\mathfrak{g}_n} s_n) = \mathfrak{f}_n$ and $\texttt{cost}_A(s_n, \mathfrak{f}_n) = \vec{u}_n$. That is, every step $s_i \xrightarrow{\mathfrak{g}_{i+1}} s_{i+1}$ in the computation λ is decomposed into two parts: $s_i \xrightarrow{\vec{u}_i} (s_i, \mathfrak{f}_i) \to (\mathfrak{g}_{i+1}, s_{i+1})$. It is worth noting that the definition of $\texttt{ext}(\lambda, F_A)$ essentially uses the set of agents A. However, in the paper, such an infinite sequence is needed only when the computation respects a strategy. That is why, we emphasize this with the notation $\texttt{ext}(\lambda, F_A)$. Similar considerations are followed in the sequel.

The computations in $\texttt{Comp}(s^\star, F_A)$ can be organised as an infinite tree that corresponds to a derivation skeleton for $\mathcal{A}_{\mathfrak{M},A,s^\star}$. Below we define an infinite tree $\mathfrak{T}_{F_A}$, a labelling function $\mathfrak{L} : \mathfrak{T}_{F_A} \to S$ and a partial map $\mathfrak{R} : \mathfrak{T}_{F_A} \times \mathfrak{T}_{F_A} \to (\bigcup_{s' \in S} D(s'))$.

- $\mathfrak{L}(\varepsilon) \overset{\text{def}}{=} s^\star$.
- For all finite words $\mathfrak{w} = k_1 \cdots k_\beta$ in $\mathfrak{T}_{F_A}$ such that $\mathfrak{L}(\mathfrak{w})$ is already defined, we add to $\mathfrak{T}_{F_A}$ the values $k_1 \cdots k_\beta \cdot 1, \ldots, k_1 \cdots k_\beta \cdot \alpha$ such that
  - $F_A(\mathfrak{L}(\varepsilon) \xrightarrow{\mathfrak{R}(\varepsilon,k_1)} \mathfrak{L}(k_1) \xrightarrow{\mathfrak{R}(k_1,k_1k_2)} \mathfrak{L}(k_1k_2) \cdots \xrightarrow{\mathfrak{R}(k_1\cdots k_{\beta-1},k_1\cdots k_\beta)} \mathfrak{L}(\mathfrak{w})) = \mathfrak{f}$
  - $\{(\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)\} = \{(\mathfrak{g}, s'') \in S \mid s'' = \delta(s', \mathfrak{g}),\ \mathfrak{g} \in D(s'),\ \mathfrak{f} \sqsubseteq \mathfrak{g}\}$ with $s' = \mathfrak{L}(\mathfrak{w})$.
  - For all $j \in [1, \alpha]$, we have $\mathfrak{L}(k_1 \cdots k_\beta \cdot j) \overset{\text{def}}{=} s_j$ and $\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot j) \overset{\text{def}}{=} \mathfrak{g}_j$.

The tree $\mathfrak{T}_{F_A}$ is defined by saturation of the above rules, and the maps $\mathfrak{L}$ and $\mathfrak{R}$ are defined accordingly. The structure $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ is a labelled transition system with a tree-like structure encoding all the infinite computations respecting the strategy F_A. A maximal branch $\mathfrak{w}$ of $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ is understood as an element of $(\mathbb{N} \setminus \{0\})^\omega$, such that any (strict) finite prefix of $\mathfrak{w}$ belongs to $\mathfrak{T}_{F_A}$. The label of $\mathfrak{w}$, written $\mathtt{lab}(\mathfrak{w})$, is defined as follows:

$$\mathtt{lab}(\mathfrak{w}) \overset{\text{def}}{=} \mathfrak{L}(\varepsilon) \xrightarrow{\mathfrak{R}(\varepsilon,k_1)} \mathfrak{L}(k_1) \xrightarrow{\mathfrak{R}(k_1,k_1k_2)} \mathfrak{L}(k_1k_2) \cdots \xrightarrow{\mathfrak{R}(k_1\cdots k_{\beta-1},k_1\cdots k_\beta)} \mathfrak{L}(k_1 \cdots k_\beta) \cdots$$

where $\mathfrak{w} = k_1k_2k_3 \cdots$. By construction, $\mathtt{lab}(\mathfrak{w})$ is a maximal computation.

Lemma 2.

(I) *For every maximal computation λ starting at $s^\star$ and respecting F_A, there is a maximal branch $\mathfrak{w}$ in $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ such that $\lambda = \mathtt{lab}(\mathfrak{w})$.*

(II) *For every maximal branch $\mathfrak{w}$ in $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$, there is a maximal computation λ starting at $s^\star$ and respecting F_A such that $\mathtt{lab}(\mathfrak{w}) = \lambda$.*

The (omitted) proof simply reflects that $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ contains all the computations from $s^\star$ that respect the strategy F_A.

We build a derivation skeleton $\mathcal{D} : \mathfrak{T}_{F_A} \to (R_1 \cup R_2)$ as follows, where all the maximal branches of $\mathfrak{T}_{F_A}$ are infinite (we therefore do not need to include $\perp$ in the range of $\mathcal{D}$).

- $\mathcal{D}(\varepsilon) = (s_0, \mathtt{cost}_A(s_0, F_A(s_0)), (s_0, F_A(s_0)))$ with $s_0 = s^\star$.
- $\mathcal{D}(1) = ((s_0, F_A(s_0)), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$ where $1, \ldots, \alpha \in \mathfrak{T}_{F_A}$ (but $\alpha + 1 \notin \mathfrak{T}_{F_A}$), for all $j \in [1, \alpha]$, $\mathfrak{T}_{F_A}(j) = s_j$ and $\mathfrak{R}(\varepsilon, j) = \mathfrak{g}_j$. By construction of $\mathcal{A}_{\mathfrak{M},A,s^\star}$, $\mathcal{D}(1)$ is the unique fork rule starting from $(s_0, F_A(s_0))$.
- Let $\mathfrak{n} = 1k_1 1 \cdots 1k_\beta 1$ with $k_1, \ldots, k_\beta \geq 1$ and such that

  $$\mathcal{D}(1k_1 1 \cdots k_\beta) = ((\mathfrak{g}, s'), \mathtt{cost}_A(s', \mathfrak{f}), (s', \mathfrak{f})) \in R_1.$$

  Then, $\mathcal{D}(\mathfrak{n}) = ((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$ where $k_1 \cdots k_\beta 1, \ldots, k_1 \cdots k_\beta \alpha \in \mathfrak{T}_{F_A}$ (but $k_1 \cdots k_\beta(\alpha + 1) \notin \mathfrak{T}_{F_A}$), for all $j \in [1, \alpha]$,
  - $\mathfrak{T}_{F_A}(k_1 \cdots k_\beta \cdot j) = s_j$ and,
  - $\mathfrak{R}(k_1 \cdots k_\beta, k_1 \cdots k_\beta j) = \mathfrak{g}_j$.

  By construction of $\mathcal{A}_{\mathfrak{M},A,s^\star}$, $\mathcal{D}(\mathfrak{n})$ is the unique fork rule starting from $(s', \mathfrak{f})$.
- Let $\mathfrak{n} = 1k_1 1 \cdots 1k_\beta$. By construction we can assume that we already have that $\mathcal{D}(1k_1 1 \cdots k_{\beta-1} 1) = ((s', \mathfrak{f}'), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha))$. Let $\mathfrak{g}'$ be equal to $\mathfrak{R}(k_1 \cdots k_\beta, k_1 \cdots k_\beta \cdot 1)$ and $\mathfrak{f}$ be the restriction of $\mathfrak{g}'$ to A (so $\mathfrak{f} \sqsubseteq \mathfrak{g}'$). Then,

  $$\mathcal{D}(\mathfrak{n}) = ((\mathfrak{g}_{k_\beta}, s_{k_\beta}), \mathtt{cost}_A(s_{k_\beta}, \mathfrak{f}), (s_{k_\beta}, \mathfrak{f})).$$

  Note that $\mathcal{D}(\mathfrak{n})$ is indeed a valid unary rule.

Given an infinite branch $\mathfrak{w}$ of $\mathcal{D}$ (resp. $\mathfrak{w}$ of the derivation $\hat{\mathcal{D}}$ based on $\mathcal{D}$), say $\mathfrak{w} = 1k_1 1k_2 1k_3 \cdots \in \mathbb{N}^\omega$, we define the extended computation $\mathtt{ext}(\mathfrak{w}, F_A)$ as follows. Suppose that the label of such a branch is characterised by the values below:

- $\mathcal{D}(\varepsilon) = (s_0, \vec{u}_0, (s_0, \mathfrak{f}_0))$; $\mathcal{D}(1) = ((s_0, \mathfrak{f}_0), (\mathfrak{g}^1_1, s^1_1), \ldots, (\mathfrak{g}^1_{\alpha_1}, s^1_{\alpha_1}))$.
- …
- $\mathcal{D}(1k_1 1 \cdots k_i) = ((\mathfrak{g}^i_{k_i}, s^i_{k_i}), \vec{u}_i, (s^i_{k_i}, \mathfrak{f}_i))$.
- $\mathcal{D}(1k_1 1 \cdots k_i 1) = ((s^i_{k_i}, \mathfrak{f}_i), (\mathfrak{g}^{i+1}_1, s^{i+1}_1), \ldots, (\mathfrak{g}^{i+1}_{\alpha_{i+1}}, s^{i+1}_{\alpha_{i+1}}))$.
- …
- $\mathcal{D}(1k_1 1 \cdots k_{\beta-1}) = ((\mathfrak{g}^{\beta-1}_{k_i}, s^{\beta-1}_{k_{\beta-1}}), \vec{u}_{\beta-1}, (s^{\beta-1}_{k_{\beta-1}}, \mathfrak{f}_{\beta-1}))$.
- $\mathcal{D}(1k_1 1 \cdots k_{\beta-1} 1) = ((s^{\beta-1}_{k_{\beta-1}}, \mathfrak{f}_{\beta-1}), (\mathfrak{g}^\beta_1, s^\beta_1), \ldots, (\mathfrak{g}^\beta_{\alpha_\beta}, s^\beta_{\alpha_\beta}))$.
- $\cdots$

Then,

$$\mathtt{ext}(\mathfrak{w}, F_A) \overset{\text{def}}{=} s_0 \xrightarrow{\vec{u}_0} (s_0, \mathfrak{f}_0) \to (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\vec{u}_1} (s^1_{k_1}, \mathfrak{f}_1) \to (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\vec{u}_2} (s^2_{k_2}, \mathfrak{f}_2) \to (\mathfrak{g}^3_{k_3}, s^3_{k_3}) \cdots$$

Lemma 3.

(I) *For every maximal computation λ starting at $s^\star$ and respecting F_A, there is a maximal branch $\mathfrak{w}$ in $\mathcal{D}$ such that* $\mathtt{ext}(\lambda, F_A) = \mathtt{ext}(\mathfrak{w}, F_A)$.
(II) *For every maximal branch $\mathfrak{w}$ in $\mathcal{D}$, there is a maximal computation λ starting at $s^\star$ and respecting F_A such that* $\mathtt{ext}(\mathfrak{w}, F_A) = \mathtt{ext}(\lambda, F_A)$.

The reduction can be easily verified. Note, however, that the proponent restriction is essential for its correctness.

Theorem 1. *There is a $\vec{b}$-strategy w.r.t. $s^\star$ in $\mathfrak{M}$ iff there is a proof in $\mathcal{A}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and every maximal branch is infinite.*

Proof. First suppose that there is a $\vec{b}$-strategy F_A w.r.t. $s^\star = s_0$ in $\mathfrak{M}$. Let us consider the structures $(\mathfrak{T}_{F_A}, \mathfrak{R}, \mathfrak{L})$ and $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2)$ as defined above. By Lemma 1, there is a unique derivation $\hat{\mathcal{D}}$ of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ based on $\mathcal{D}$ such that $\hat{\mathcal{D}}(\varepsilon) = (s^\star, \vec{b})$. It remains to show that $\hat{\mathcal{D}}$ is indeed a proof. Let $\mathfrak{w} = 1k_1 1k_2 1 \cdots$ be a maximal branch of $\hat{\mathcal{D}}$ (we use the previous notations about $\mathcal{D}$ such as those about the $\vec{u}_i$'s). We have:

- $\hat{\mathcal{D}}(\varepsilon) = (s_0, \vec{b})$.
- $\hat{\mathcal{D}}(1) = ((s_0, \mathfrak{f}_0), \vec{u}_0 + \vec{b})$ with $\mathcal{D}(\varepsilon) = (s_0, \vec{u}_0, (s_0, \mathfrak{f}_0))$.
- ...
- $\hat{\mathcal{D}}(1k_1 1 \cdots k_i) = ((\mathfrak{g}^i_{k_i}, s^i_{k_i}), \sum_{j=1}^{i-1} \vec{u}_j + \vec{b})$ with
$$\mathcal{D}(1k_1 1 \cdots k_{i-1} 1) = ((s^{i-1}_{k_{i-1}}, \mathfrak{f}_{i-1}), (\mathfrak{g}^i_1, s^i_1), \ldots, (\mathfrak{g}^i_{\alpha_i}, s^i_{\alpha_i})).$$
- $\hat{\mathcal{D}}(1k_1 1 \cdots k_i 1) = ((s^i_{k_i}, \mathfrak{f}_i), \sum_{j=1}^{i} \vec{u}_j + \vec{b})$ with $\mathcal{D}(1k_1 1 \cdots k_i) = ((\mathfrak{g}^i_{k_i}, s^i_{k_i}), \vec{u}_i, (s^i_{k_i}, \mathfrak{f}_i))$.
- $\cdots$

By Lemma 3, there is a maximal computation λ starting at $s^\star$ and respecting F_A such that $\mathtt{ext}(\mathfrak{w}, F_A) = \mathtt{ext}(\lambda, F_A)$. Since F_A is a $\vec{b}$-strategy, λ is $\vec{b}$-consistent and therefore for all $i \geq 0$, we have $\vec{0} \preceq \sum_{j=1}^{i-1} \vec{u}_j + \vec{b}$, which implies that $\hat{\mathcal{D}}$ is a proof.

For the proof of the other direction, assuming that there is a proof $\hat{\mathcal{D}}$ whose root is labelled by $(s^\star, \vec{b})$ and every maximal branch is infinite, we can extract from the underlying derivation $\mathcal{D}$ a strategy F_A (see the similar construction in the proof of Theorem 3 below). Lemma 3 and the fact that $\hat{\mathcal{D}}$ is admissible entail that F_A is a $\vec{b}$-strategy w.r.t. $s^\star$ (details are omitted). □

Transitions in $\mathfrak{M}$ can be defined as triples $(s', \mathfrak{g}, s'')$ such that $\delta(s', \mathfrak{g}) = s''$. A transition is also denoted by the expression $s' \xrightarrow{\mathfrak{g}} s''$. The set of transitions of $\mathfrak{M}$ is denoted by $\Sigma_{\mathfrak{M}}$. It is interpreted as a finite alphabet when $\mathfrak{M}$ is finite. An infinite computation $\lambda = s_0 \xrightarrow{\mathfrak{g}_1} s_1 \xrightarrow{\mathfrak{g}_2} s_2 \ldots$ can be equivalently represented by the ω-word in $\Sigma^\omega_{\mathfrak{M}}$ (with contiguous transitions)

$$(s_0 \xrightarrow{\mathfrak{g}_1} s_1) \cdot (s_1 \xrightarrow{\mathfrak{g}_2} s_2) \cdot (s_2 \xrightarrow{\mathfrak{g}_3} s_3) \cdots$$

An ω-word $\mathfrak{w} \in \Sigma^\omega_{\mathfrak{M}}$ is said *to be with contiguous transitions* whenever at any position, the second state of the transition is equal to the first state of the next position.

Given an infinite branch of the proof corresponding to the extended computation

$$s \xrightarrow{\vec{u}_0} (s, \mathfrak{f}_0) \to (\mathfrak{g}^1_{k_1}, s^1_{k_1}) \xrightarrow{\vec{u}_1} (s^1_{k_1}, \mathfrak{f}_1) \to (\mathfrak{g}^2_{k_2}, s^2_{k_2}) \xrightarrow{\vec{u}_2} (s^2_{k_2}, \mathfrak{f}_2) \to (\mathfrak{g}^3_{k_3}, s^3_{k_3}) \cdots$$

its $\Sigma_{\mathfrak{M}}$*-projection* is defined as the sequence

$$(s \xrightarrow{\mathfrak{g}^1_{k_1}} s^1_{k_1}) \cdot (s^1_{k_1} \xrightarrow{\mathfrak{g}^2_{k_2}} s^2_{k_2}) \cdot (s^2_{k_2} \xrightarrow{\mathfrak{g}^3_{k_3}} s^3_{k_3}) \cdots$$

Lemma 4. *Let $\mathrm{L} \subseteq \Sigma^\omega_{\mathfrak{M}}$ and $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$. The statements below are equivalent.*

1. *There is a $\vec{b}$-strategy F_A w.r.t. $s^\star$ in $\mathfrak{M}$ such that the set of computations* $\mathtt{Comp}(s^\star, F_A)$ *is included in* L.
2. *There is a proof in $\mathcal{A}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$, every maximal branch is infinite and its $\Sigma_{\mathfrak{M}}$-projection belongs to* L.

Lemma 4 is a consequence of Theorem 1, and Lemma 3 is key to establishing formal correspondences between $\mathfrak{M}$ and $\mathcal{A}_{\mathfrak{M},A,s^\star}$. The main challenge is to determine classes of languages for which decidability can be obtained by using only the decidability (and complexity characterisation) of the state reachability and non-termination problems for AVASS.

Given $S' \subseteq S$ with $s^\star \in S'$, let $\mathrm{L}_{S'} \subseteq \Sigma^\omega_{\mathfrak{M}}$ be the set of all ω-words such that the transitions use only states in S'. Let us define $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ as a restriction of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ in which the unary and fork rules have no way to go out of S'. Alternatively, $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ can be understood as the restriction of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ to rules that only involve states in S' (assuming that $s^\star$ is already in S'). We define $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star} \stackrel{\text{def}}{=} (Q, r, R_1, R_2)$ as follows:

$$Q \stackrel{\text{def}}{=} \{s^\star\} \cup \{(s', \mathfrak{f}) \mid s' \in S',\ \mathfrak{f} \in D_A(s')\} \cup \{(\mathfrak{g}, s') \mid s', s'' \in S',\ \mathfrak{g} \in D(s''),\ \delta(s'', \mathfrak{g}) = s'\}.$$

- The set of unary rules R_1 contains the following elements.
  - For all $\mathfrak{f} \in D_A(s^\star)$, $(s^\star, \mathtt{cost}_A(s^\star, \mathfrak{f}), (s^\star, \mathfrak{f}))$.
  - For all $(\mathfrak{g}, s') \in Q$, for all $\mathfrak{f} \in D_A(s')$, $((\mathfrak{g}, s'), \mathtt{cost}_A(s', \mathfrak{f}), (s', \mathfrak{f}))$.
- The set of fork rules R_2 contains the following elements.
  - For all $(s', \mathfrak{f}) \in Q$, let $\{(\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)\} = \{(\mathfrak{g}, s'') \mid s'' = \delta(s', \mathfrak{g}),\ \mathfrak{g} \in D(s'),\ \mathfrak{f} \sqsubseteq \mathfrak{g}\}$.
    If $\{s_1, \ldots, s_\alpha\} \subseteq S'$, then we add the α-ary fork rule

    $$((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)).$$

    (Otherwise, nothing is added.)

  So, there is at most one fork rule starting from $(s', \mathfrak{f})$ (possibly zero).

Lemma 5. *Assuming that $s^\star \in S'$, the statements below are equivalent.*

1. *There is a $\vec{b}$-strategy F_A w.r.t. $s^\star$ in $\mathfrak{M}$ such that the set of computations $\mathtt{Comp}(s^\star, F_A)$ only visit states in S'.*
2. *There is a proof in $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and every maximal branch is infinite (a positive instance of the non-termination problem for AVASS).*

Note that the way $\vec{b}$-strategies are defined, in Lemma 5(1), F_A generates maximal and infinite computations in which only states in S' are visited. Similarly, the proof in Lemma 5(2) contains only maximal and infinite branches and its root is precisely $(s^\star, \vec{b})$.

Proof. Let $\mathcal{A}_{\mathfrak{M},A,s^\star} = (Q, r, R_1, R_2)$ and $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star} = (Q', r, R_1', R_2')$. By construction, we have $Q' \subseteq Q$, $R_1' \subseteq R_1$ and $R_2' \subseteq R_2$. We write $\Sigma'_{\mathfrak{M}}$ to denote the alphabet $\{s_1 \xrightarrow{\mathfrak{g}} s_2 : s_1, s_2 \in S'$ and*and* $\delta(s_1, \mathfrak{g}) = s_2\}$ and L to denote the ω-regular language in $(\Sigma'_{\mathfrak{M}})^\omega$ made of infinite sequences of contiguous transitions such that only states in S' can occur.

(1) → (2). Suppose there is a $\vec{b}$-strategy F_A w.r.t. $s^\star$ in $\mathfrak{M}$ such that the set of computations $\mathtt{Comp}(s^\star, F_A)$ only visit states in S', which amounts to having the set of computations included in L. By Lemma 4, there is a proof in $\mathcal{A}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$, every maximal branch is infinite and it belongs to L. Since $s^\star \in S'$, all the rules in $R_1' \cup R_2'$ only involve states in S' and the above proof only visits such states, we have that there is a proof in $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and every maximal branch is infinite.

(2) → (1). Suppose that there is a proof in $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and every maximal branch is infinite. Since $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ is defined as a restriction of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ with $Q' \subseteq Q$, $R_1' \subseteq R_1$ and $R_2' \subseteq R_2$, there is also a proof in $\mathcal{A}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$, every maximal branch is infinite and only states in S' are visited. Equivalently, there is a proof in $\mathcal{A}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$, every maximal branch is infinite and it belongs to L. By Lemma 4, there is a $\vec{b}$-strategy w.r.t. $s^\star$ in $\mathfrak{M}$ such that the set of computations $\mathtt{Comp}(s^\star, F_A)$ is included in L, hence only states in S' are visited. □

Lemma 5 is useful to handle formulae of the form $\langle\langle A^{\vec{b}}\rangle\rangle \mathsf{G}\phi$. Let us consider a similar treatment that will be useful to handle formulae of the form $\langle\langle A^{\vec{b}}\rangle\rangle \phi_1 \mathsf{U} \phi_2$. Given $S_1, S_2 \subseteq S$ with $s^\star \in S_1 \cup S_2$, let L_{S_1,S_2} be the set of all ω-words with contiguous transitions such that the projection under S belongs to $S_1^* \cdot S_2 \cdot S^\omega$. As usual, the projection of $(s_0 \xrightarrow{\mathfrak{g}_1} s_1) \cdot (s_1 \xrightarrow{\mathfrak{g}_2} s_2) \cdot (s_2 \xrightarrow{\mathfrak{g}_3} s_3) \cdots$ under S is understood as $s_0 s_1 s_2 s_3 \cdots$.

Lemma 6. *The statements below are equivalent:*

1. *There is a $\vec{b}$-strategy w.r.t. $s^\star$ in $\mathfrak{M}$ such that $\mathtt{Comp}(s^\star, F_A) \subseteq \mathrm{L}_{S_1,S_2}$.*
2. *There is a finite proof in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and each leaf contains a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s^\star\})$ (a positive instance of the state reachability problem for AVASS).*

Note that $S_2 \cap \{s^\star\}$ is $\{s^\star\}$ if $s^\star \in S_2$ and $\emptyset$ otherwise. The proof of Lemma 6 below relies on the fact that, in resource-bounded concurrent game structures, $\mathtt{idle} \in \mathtt{act}(s, a)$ for all agents a and states s.

Proof. The proof is similar to the proof of Lemma 5 except that we need to relate finite proofs to infinite ones, and in doing so we take advantage of the presence of the idle action in concurrent game structures.

Let $\mathcal{A}_{\mathfrak{M},A,s^\star} = (Q, r, R_1, R_2)$ and $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s^\star} = (Q', r, R'_1, R'_2)$. By construction, we have $Q' \subseteq Q$, $R'_1 \subseteq R_1$ and $R'_2 \subseteq R_2$.

(1) $\rightarrow$ (2). Suppose there is a $\vec{b}$-strategy F_A w.r.t. $s^\star$ in $\mathfrak{M}$ such that the computations in $\mathtt{Comp}(s^\star, F_A)$ visit a state in S_1 until a state in S_2 is visited, which amounts to having the set of computations included in L_{S_1,S_2}. By Lemma 4, there is a proof $\hat{\mathcal{D}}$ in $\mathcal{A}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s, \vec{b})$, every maximal branch is infinite and it belongs to L_{S_1,S_2}. Let $\hat{\mathcal{D}}'$ be the finite proof obtained from $\hat{\mathcal{D}}$ by pruning any subtree as soon as a node is labelled by a control state in S_2. Existence of such a finite proof is guaranteed by König's Lemma. It is easy to check that $\hat{\mathcal{D}}'$ is a finite proof in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and each leaf contains a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s^\star\})$.

(2) $\rightarrow$ (1). Suppose that there is a finite proof $\hat{\mathcal{D}}$ in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s^\star}$ whose root is labelled by $(s^\star, \vec{b})$ and each leaf contains a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s^\star\})$. One can extend $\hat{\mathcal{D}}$ in order to obtain an infinite proof $\hat{\mathcal{D}}'$ such that every maximal branch is infinite and it belongs to L_{S_1,S_2}. Any leaf labelled by the control state $(\mathfrak{g}, s')$ is further extended by application of the unary rule $(\mathfrak{g}, s') \xrightarrow{\vec{0}} (s', \mathfrak{f})$ where $\mathfrak{f}$ is the idle joint action (with the control state $s^\star$, a similar method applies). Similarly, any leaf labelled by the control state $(s', \mathfrak{f})$ is further extended by application of the unique fork rule starting by $(s', \mathfrak{f})$. It is easy to check that this not only leads to a derivation but also to a proof, because the extension only deals with the update vector $\vec{0}$. By Lemma 4, there is a $\vec{b}$-strategy w.r.t. $s^\star$ in $\mathfrak{M}$ such that the set of computations $\mathtt{Comp}(s^\star, F_A)$ is included in L_{S_1,S_2}. □

4.2. 2EXPTIME upper bound

The upper bound is established by giving a labelling algorithm as done in [5] or for standard temporal logics such as CTL and CTL*. The main difference with [5] is that the treatment of the cases with strategy modalities is not performed in an ad-hoc fashion using the fact that $(\mathbb{N}^r, \preceq)$ is a well-quasi-ordering by Dickson's Lemma [22] but rather we explicitly call subroutines that solve decision problems on AVASS. The existence of such subroutines is due to [40] for *monotonic games*, and their complexity upper bounds are due to [33, Theorem 3.4] and [17, Theorem 3.1]. The proof of the 2EXPTIME upper bound is divided into three main steps:

1. we introduce a slight extension of AVASS such that the decision problems remain in 2EXPTIME;
2. we show that the cases for the strategy modalities can be faithfully reduced to subroutines for problems on such extended AVASS (a consequence of developments from Section 4.1);
3. finally, we design a labelling algorithm and establish the complexity upper bound from it.

First, let us introduce a slight extension of decision problems for AVASS.

Lemma 7. *In the following extension of AVASS the state reachability and non-termination problems remain in 2EXPTIME:*

- *Fork rules can be α-ary for any $\alpha \geq 1$ (but there is only a finite number of them).*
- *Reachability is related to a subset $Q_f \subseteq Q$ (instead of a singleton set).*
- *The initial configuration is $(q_0, \vec{b})$ with $\vec{b} \in \mathbb{N}^r$ instead of the fixed tuple $\vec{0}$.*
- *The value ω in $\vec{b}$ is allowed and absorbs any other value in $\mathbb{Z}$ (a means to ignore components, i.e. to reduce the dimension).*

The proof is fairly standard, and consists in using Proposition 2 by simulating a non-binary fork by a linear-size gadget made of unary and binary forking rules, and by adding binary forking rules from states in Q_f to a new single final state.

Proof. The lemma states four ways to extend the decision problems on AVASS, either by slightly extending the notion of AVASS, or by considering more general inputs for the problems. For each extension, we show how this can be encoded into the state reachability and the non-termination problems on AVASS using only polynomial-time reductions. The proof of the lemma is then obtained by composition of the reductions (polynomial-time reductions are also known to be closed under compositions) and by invoking Proposition 2 to get the 2EXPTIME upper bound.

- Let $\mathcal{A} = (Q, r, R_1, R_2)$ be an alternating VASS, $q_0, q_f \in Q$, and $\mathfrak{r} = (q_1, \ldots, q_{\alpha+1})$ be an (extended) α-ary rule. If $\alpha = 1$, the rule can be treated as a unary rule with the update vector $\vec{0}$, whereas if $\alpha = 2$, it can be treated as a standard binary fork rule. So assume $\alpha \geq 3$. Let R'_2 be the following set of binary fork rules derived from $\mathfrak{r}$ where $q'_2, \ldots, q'_{\alpha-1}$ are new control states:

$$R'_2 = \{(q_1, q_2, q'_2)\} \cup \{(q'_j, q_{j+1}, q'_{j+1}) \mid j \in [2, \alpha - 2]\} \cup \{(q'_{\alpha-1}, q_\alpha, q_{\alpha+1})\}.$$

It is easy to show that the statements below are equivalent:
– there is a finite proof in $(Q, r, R_1, R_2 \uplus \{\mathfrak{r}\})$ whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$;
– there is a finite proof in $(Q \uplus \{q'_j \mid j \in [2, \alpha-1]\}, r, R_1, R_2 \uplus R'_2)$ whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.

If there is more than one extended fork rule, we apply the above reduction as many times as necessary, leading eventually to a reduction to an instance of the state reachability problem for AVASS. The same reduction also works for the non-termination problem.

- Let $\mathcal{A} = (Q, r, R_1, R_2)$, $q_0 \in Q$ and $Q_f = \{q_1, \ldots, q_\beta\}$.
  Let $\mathcal{A}' = (Q \uplus \{q_f^{\text{new}}\}, r, R'_1, R_2)$ be defined from $\mathcal{A}$ such that

$$R'_1 \overset{\text{def}}{=} R_1 \uplus \{q_i \xrightarrow{\vec{0}} q_f^{\text{new}} \mid i \in [1, \beta]\}.$$

  It is easy to show that the statements below are equivalent:
  – there is a finite proof in $\mathcal{A}$ whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $Q_f \times \mathbb{N}^r$;
  – there is a finite proof in $\mathcal{A}'$ whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $\{q_f^{\text{new}}\} \times \mathbb{N}^r$.

- Let $\mathcal{A} = (Q, r, R_1, R_2)$, $q_0, q_f \in Q$ and $\vec{b} \in \mathbb{N}^r$ and $\mathcal{A}' = (Q \uplus \{q'_0\}, r, R_1 \uplus \{q'_0 \xrightarrow{\vec{b}} q_0\}, R_2)$. It is easy to show that the statements below are equivalent:
  – there is a finite proof in $\mathcal{A}$ whose root is equal to $(q_0, \vec{b})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$;
  – there is a finite proof in $\mathcal{A}'$ whose root is equal to $(q'_0, \vec{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$.
  Similarly, the statements below are equivalent:
  – there is a proof in $\mathcal{A}$ whose root is equal to $(q_0, \vec{b})$ and all the maximal branches are infinite;
  – there is a proof in $\mathcal{A}'$ whose root is equal to $(q'_0, \vec{0})$ and all the maximal branches are infinite.

- Let $\mathcal{A} = (Q, r, R_1, R_2)$, $q_0, q_f \in Q$ and $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$. We are looking for proofs whose root is labelled by $(q_0, \vec{b})$ and any occurrence of ω in $\vec{b}$ remains in the proof for all the descendant nodes, which amounts to ignoring some components. Indeed, if ω occurs in a component at the root of the proof, the value ω is propagated to all descendant nodes in that component.
  Suppose that ω occurs at least once in $\vec{b}$ and let $\{i_1, \ldots, i_\beta\} \subseteq [1, r]$ be the set of positions where ω occurs in $\vec{b}$. Let $\gamma = r - \beta$ and let $j_1 < \cdots < j_\gamma$ be the indices in $[1, r] \setminus \{i_1, \ldots, i_\beta\}$. Let $\mathfrak{h} : [1, \gamma] \to \{j_1, \cdots, j_\gamma\}$ be the bijection such that $\mathfrak{h}(n) \overset{\text{def}}{=} j_n$. We define the alternating VASS $\mathcal{A}' = (Q, \gamma, R'_1, R_2)$ obtained from $\mathcal{A}$ by removing the components in positions in $\{i_1, \ldots, i_\beta\}$. The map $\mathfrak{h}$ is extended to $\bar{\mathfrak{h}} : (\mathbb{Z} \cup \{\omega\})^r \to \mathbb{Z}^\gamma$ such that for all $\vec{u} \in \mathbb{Z}^r$, $n \in [1, \gamma]$ we have $\bar{\mathfrak{h}}(\vec{u})(n) \overset{\text{def}}{=} \vec{u}(j_n)$. The set of unary rules R'_1 is defined from R_1 as follows:

$$R'_1 \overset{\text{def}}{=} \{q \xrightarrow{\bar{\mathfrak{h}}(\vec{u})} q' \mid q \xrightarrow{\vec{u}} q' \in R_1\}.$$

  It is easy to show that the statements below are equivalent:
  – there is a finite proof in $\mathcal{A}$ whose root is equal to $(q_0, \vec{b})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$;
  – there is a finite proof in $\mathcal{A}'$ whose root is equal to $(q_0, \bar{\mathfrak{h}}(\vec{b}))$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^\gamma$.
  The same reduction also works for the non-termination problem. □

Lemma 8 below relates the satisfaction of a formula with the outermost connective $\langle\langle A^{\vec{b}}\rangle\rangle\mathsf{U}$ and the state reachability problem for AVASS.

Lemma 8. *The statements below are equivalent:*

(I) $\mathfrak{M}, s^\star \models \langle\langle A^{\vec{b}}\rangle\rangle \phi_1 \mathsf{U} \phi_2$.
(II) *there is a finite proof in* $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M}, A, s^\star}$ *whose root is equal to* $(s^\star, \vec{b})$ *and each leaf has a control state in* $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s^\star\})$ *with* $S_i = \{s' \mid \mathfrak{M}, s' \models \phi_i\}$, $i \in \{1, 2\}$.

This is a direct consequence of Lemma 6.

Lemma 9 relates to the satisfaction of a formula with outermost connective $\langle\langle A^{\vec{b}}\rangle\rangle\mathsf{G}$ and the non-termination problem for AVASS.

Lemma 9. *Assuming that* $\mathfrak{M}, s^\star \models \phi_1$, *the statements below are equivalent:*

(I) $\mathfrak{M}, s^\star \models \langle\langle A^{\vec{b}}\rangle\rangle \mathsf{G} \phi_1$.

(II) *there is a proof in* $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$ *with* $S' = \{s' \mid \mathfrak{M}, s' \models \phi_1\}$ *whose root is equal to* $(s^\star, \vec{b})$ *and every maximal branch is infinite.*
(III) $\mathcal{A}^{S'}_{\mathfrak{M},A,s^\star}$, $(s^\star, \vec{b})$ *is a positive instance of the non-termination problem for AVASS.*

This is a direct consequence of Lemma 5.

Theorem 2. *The model checking problem for RB±ATL is in 2*EXPTIME.

Algorithm 1 – RB±ATL model checking –

```
 1: procedure GMC($\mathfrak{M}, \phi$)
 2:     case $\phi$ of
 3:     $p$:  return $\{s \in S \mid s \in Lab(p)\}$
 4:     $\neg\psi$:  return $S \setminus$ GMC($\mathfrak{M}, \psi$)
 5:     $\psi_1 \wedge \psi_2$:  return GMC($\mathfrak{M}, \psi_1$) $\cap$ GMC($\mathfrak{M}, \psi_2$)
 6:     $\langle\langle A^{\vec{b}}\rangle\rangle \mathsf{X}\psi$:  return $\{s \mid \exists\, \mathfrak{f} \in D_A(s), \vec{0} \preceq \mathtt{cost}_A(s, \mathfrak{f}) + \vec{b}$, for all $\mathfrak{f} \sqsubseteq \mathfrak{g} \in D(s)$, $\delta(s, \mathfrak{g}) \in$ GMC($\mathfrak{M}, \psi$)$\}$
 7:     $\langle\langle A^{\vec{b}}\rangle\rangle \mathsf{G}\psi$:  $S_1 :=$ GMC($\mathfrak{M}, \psi$);
         return $\{s^\star \in S_1 \mid \mathcal{A}^{S_1}_{\mathfrak{M},A,s^\star}, (s^\star, \vec{b})$ is a positive instance of the non-termination problem$\}$.
 8:     $\langle\langle A^{\vec{b}}\rangle\rangle \psi_1 \mathsf{U} \psi_2$:  return $\{s^\star \mid \mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s^\star}, (s, \vec{b}), S'_2$ is a positive instance of the state reachability problem$\}$ with $S_1 =$ GMC($\mathfrak{M}, \psi_1$), $S_2 =$
    GMC($\mathfrak{M}, \psi_2$), $S'_2 = \{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s^\star\})$
 9:     end case
10: end procedure
```

Proof. Algorithm 1 is a global model checking algorithm that takes as input a resource-bounded concurrent game structure $\mathfrak{M}$ and a formula ϕ (both built on the same set of agents and with the same number of resources) and returns the set of states that satisfies the formula. By structural induction, one can show that GMC$(\mathfrak{M}, \psi) = \{s \in S \mid \mathfrak{M}, s \models \psi\}$ by using Lemma 8 and Lemma 9. We use the fact that the state reachability and the non-termination problems for extended AVASS are decidable by [40] and by Lemma 7 (for the extension). Let us show how the proof by induction works.

Case $\psi = \langle\langle A^{\vec{b}}\rangle\rangle \mathsf{G}\psi'$
The statements below are equivalent:

- $\mathfrak{M}, s \models \psi$.
- there is a $\vec{b}$-strategy F_A w.r.t. s such that the computations in $\mathtt{Comp}(s, F_A)$ only visit states in $S'_1 = \{s' \mid \mathfrak{M}, s' \models \psi'\}$ (by definition of $\models$).
- $s \in S'_1$ and there is a proof in $\mathcal{A}^{S'_1}_{\mathfrak{M},A,s}$ whose root is equal to $(s, \vec{b})$ and every maximal branch is infinite (by Lemma 9).
- $s \in S_1$ and there is a proof in $\mathcal{A}^{S_1}_{\mathfrak{M},A,s}$ whose root is equal to $(s, \vec{b})$ and every maximal branch is infinite with $S_1 =$ GMC$(\mathfrak{M}, \psi')$ (by induction hypothesis).
- $s \in S_1$ and $\mathcal{A}^{S_1}_{\mathfrak{M},A,s}$, $(s, \vec{b})$ is a positive instance of the non-terminating problem for AVASS (by definition).

Case $\psi = \langle\langle A^{\vec{b}}\rangle\rangle \psi_1 \mathsf{U} \psi_2$. The statements below are equivalent:

- $\mathfrak{M}, s \models \psi$.
- there is a $\vec{b}$-strategy F_A w.r.t. s such that for all $\lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \mathtt{Comp}(s, F_A)$, there is some $i < |\lambda|$ such that $\mathfrak{M}, s_i \models \psi_2$ and for all $j \in [0, i-1]$, we have $\mathfrak{M}, s_j \models \psi_1$.
- there is a finite proof in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$ whose root is equal to $(s, \vec{b})$ and each leaf has a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s\})$ with $S_i = \{s \mid \mathfrak{M}, s \models \psi_i\}$, $i \in \{1, 2\}$ (by Lemma 8).
- there is a finite proof in $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$ whose root is equal to $(s, \vec{b})$ and each leaf has a control state in $\{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s\})$ with $S_i =$ GMC$(\mathfrak{M}, \psi_i)$ $i \in \{1, 2\}$ (by the induction hypothesis).
- $\mathcal{A}^{S_1 \cup S_2}_{\mathfrak{M},A,s}$, $(s, \vec{b})$, S'_2 is a positive instance of the state reachability problem for AVASS with $S_i =$ GMC$(\mathfrak{M}, \psi_i)$ $i \in \{1, 2\}$ and $S'_2 = \{(\mathfrak{g}, s') \in Q \mid s' \in S_2\} \cup (S_2 \cap \{s\})$.

Case $\psi = \langle\langle A^{\vec{b}}\rangle\rangle \mathsf{X}\psi'$
The statements below are equivalent:

- $\mathfrak{M}, s \models \psi$.
- there is a $\vec{b}$-strategy F_A w.r.t. s such that for all $s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \mathtt{Comp}(s, F_A)$, we have $\mathfrak{M}, s_1 \models \psi'$ (by definition of $\models$).
- there is a $\vec{b}$-strategy F_A w.r.t. s such that for all $s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \mathtt{Comp}(s, F_A)$, we have $\mathfrak{M}, s_1 \models \psi'$, and for any finite computation extending $s_0 \xrightarrow{\mathfrak{g}_0} s_1$, F_A returns the constant map $\mathtt{idle}$ (thanks to the properties of the action $\mathtt{idle}$).

- there is $\mathfrak{f} \in D_A(s)$ such that for all $s_1 \in \mathtt{out}(s, \mathfrak{f})$, we have $\mathfrak{M}, s_1 \models \psi'$ and $\vec{0} \preceq \mathtt{cost}_A(s, \mathfrak{f}) + \vec{b}$.
- there is $\mathfrak{f} \in D_A(s)$ such that for all $\mathfrak{g} \sqsupseteq \mathfrak{f}$, we have $\mathfrak{M}, \delta(s, \mathfrak{g}) \models \psi'$ and $\vec{0} \preceq \mathtt{cost}_A(s, \mathfrak{f}) + \vec{b}$.
- there is $\mathfrak{f} \in D_A(s)$ such that for all $\mathfrak{g} \sqsupseteq \mathfrak{f}$, we have $\delta(s, \mathfrak{g}) \in \mathrm{GMC}(\mathfrak{M}, \psi')$ and $\vec{0} \preceq \mathtt{cost}_A(s, \mathfrak{f}) + \vec{b}$ (by induction hypothesis).

As far as complexity is concerned, GMC$(\mathfrak{M}, \psi)$ can be solved by using a recursion depth that is linear in the size of ψ, and the state reachability and the non-termination problems for AVASS can be solved in 2EXPTIME by [33, Theorem 3.4] and [17, Theorem 3.1]. Note also the instances of such problems can be built in polynomial time in the respective sizes of $\mathfrak{M}$ and ϕ. Consequently, the model checking problem for RB±ATL is in 2EXPTIME. □

4.3. 2EXPTIME-hardness

In this section, we show a 2EXPTIME-hardness result by reduction from the state reachability problem for AVASS. This improves the EXPSPACE-hardness result in [5].

Theorem 3. *The model checking problem for RB±ATL is 2EXPTIME-hard.*

Proof. The proof is by reduction from the state reachability problem for AVASS (see Proposition 2 or [17, Theorem 4.1]). It is divided into three main parts.

(1) We consider a restriction of the state reachability problem for AVASS that remains 2EXPTIME-hard but that simplifies the definitions in the second part of the proof. Roughly speaking, the set of control states is divided in two disjoint sets, one from which unary rules start, and the other one from which fork rules start.
(2) We then define the reduction from the restriction, taking care of the details of the resource-bounded concurrent game structures; essentially, we follow ideas similar to those in the proof of [5, Lemma 6].
(3) Finally, we establish the correctness of the reduction.

(1) Given an instance $\mathcal{A} = (Q, r, R_1, R_2)$, q_0, q_f of the state reachability problem, we further assume that there is a partition $Q = Q_1 \uplus Q_2$ such that

- $q_0, q_f \in Q_1$,
- $R_1 \subseteq Q_1 \times \mathbb{Z}^r \times Q_2$ and $R_2 \subseteq Q_2 \times Q_1 \times Q_1$,
- there is no rule starting from q_f and there is at least one unary rule starting from q_0.

The strict alternation between the control states in Q_1 and those in Q_2 can be obtained by duplicating the control states (in case a control state can start both a unary rule and a fork rule), and by adding new intermediate rules to enforce the alternation. In order to have no rule from q_f, it is sufficient to duplicate it, leading to the new state q'_f. So, q'_f behaves now as q_f in the original AVASS and in the new AVASS, no rule starts by q_f. The details follow.

Let $\mathcal{A} = (Q, r, R_1, R_2)$ be an alternating VASS. Without loss of generality, we can assume that no rule starts from q_f. Otherwise, we can introduce a new control state q_f^{new} that behaves almost as q_f: copy all the rules where q_f occurs in second or in third position by replacing q_f by q_f^{new} but no rule starting from q_f is copied (details are omitted). We can guarantee that ($\star$) there is a finite proof whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$ iff with the new AVASS there is a finite proof whose root is equal to $(q_0, \vec{0})$ and each leaf belongs to $\{q_f^{\mathrm{new}}\} \times \mathbb{N}^r$. Similarly, without loss of generality, we can assume that there is a rule in R_1 that starts from q_0. Otherwise, we add the dummy unary rule $(q_0, \vec{0}, q_0)$.

Let us build the alternating VASS $\mathcal{A}' = (Q', r, R'_1, R'_2)$ verifying the above conditions with $Q' = Q'_1 \uplus Q'_2$ such that ($\star$) iff there is a finite proof whose root is equal to $((q_0, 1), \vec{0})$ and each leaf belongs to $\{(q_f, 1)\} \times \mathbb{N}^r$. The set Q' is a subset of $Q \times \{1, 2\}$ defined by the clauses below plus auxiliary states introduced with the definition for rules:

- $(q, 1) \in Q' \overset{\mathrm{def}}{\Leftrightarrow}$ there is a rule in R_1 that starts from q or $q = q_f$. So $(q_0, 1), (q_f, 1) \in Q'$.
- $(q, 2) \in Q' \overset{\mathrm{def}}{\Leftrightarrow}$ there is a rule in R_2 that starts from q.
- $Q'_1 \supseteq \{(q, i) \in Q' \mid i = 1\}$ and $Q'_2 \supseteq \{(q, i) \in Q' \mid i = 2\}$ (the inclusions are in the right direction since Q'_1 and Q'_2 may contain auxiliary states). Obviously, $(q_0, 1), (q_f, 1) \in Q'_1$.

We now define the sets of rules R'_1 and R'_2.

- For all $q \xrightarrow{\vec{u}} q' \in R_1$ such that $(q', 2) \in Q'$, we add the rule $(q, 1) \xrightarrow{\vec{u}} (q', 2)$ to R'_1.
- For all $(q_1, q_2, q_3) \in R_2$ such that $(q_2, 1), (q_3, 1) \in Q'$, we add the fork rule

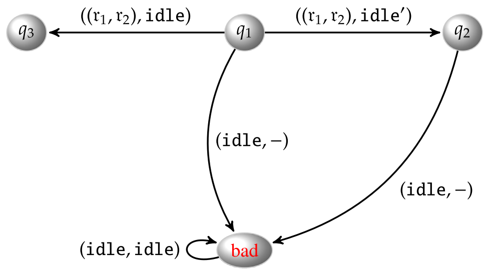
q_3
$((\mathfrak{r}_1,\mathfrak{r}_2),\mathtt{idle})$
q_1
$((\mathfrak{r}_1,\mathfrak{r}_2),\mathtt{idle}')$
q_2
$(\mathtt{idle},-)$
$(\mathtt{idle},-)$
$(\mathtt{idle},\mathtt{idle})$
bad

Fig. 3. Partial description of $\mathfrak{M}$.

$$((q_1,2),(q_2,1),(q_3,1))$$

to R_2'.

- For all $\mathfrak{r} = q \xrightarrow{\vec{u}} q' \in R_1$ such that $(q',1) \in Q'$ (alternation needs to be enforced), we add the rules below: $(q,1) \xrightarrow{\vec{u}} q^{\mathrm{new}} \in R_1'$ (q^{new} is new and depends on $\mathfrak{r}$) and $(q^{\mathrm{new}},(q',1),(q',1)) \in R_2'$. Moreover, $q^{\mathrm{new}} \in Q_2'$. This amounts to adding an intermediate fork rule leading twice to $(q',1)$ in order to guarantee that $R_1' \subseteq Q_1' \times \mathbb{Z}^r \times Q_2'$.
- For all $\mathfrak{r} = (q_1,q_2,q_3) \in R_2$ such that either $(q_2,2) \in Q'$ or $(q_3,2) \in Q'$ (again, alternation needs to be enforced). If $(q_2,2),(q_3,2) \in Q'$, then we add the rules below:
  - $((q_1,2),q_2^{\mathrm{new}},q_3^{\mathrm{new}}) \in R_2'$ where q_2^{new} and q_3^{new} are new and depend on the fork rule $\mathfrak{r}$. Moreover, these two new control states belong to Q_1'.
  - $q_2^{\mathrm{new}} \xrightarrow{\vec{0}} (q_2,2)$ and $q_3^{\mathrm{new}} \xrightarrow{\vec{0}} (q_3,2)$ belong to R_1'.

  Again, we add intermediate unary rules in order to guarantee that $R_2' \subseteq Q_2' \times Q_1' \times Q_1'$. If $(q_2,2)$, $(q_3,1)$ belong to Q' or if $(q_2,1)$, $(q_3,2)$ belong to Q', the above construction can be easily adapted.

One can show that $\mathcal{A}' = (Q',r,R_1',R_2')$ satisfies the above assumption and $(\star)$ iff there is a finite proof whose root is equal to $((q_0,1),\vec{0})$ and each leaf belongs to $\{(q_f,1)\} \times \mathbb{N}^r$.

(2) Given an instance $\mathcal{A} = (Q,r,R_1,R_2)$, q_0 and q_f with the restriction above, we build the game structure $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, Lab)$ with $Q_1 \subseteq S$ such that $\mathfrak{M}, q_0 \models \langle\langle \{1\}^{\vec{0}} \rangle\rangle \top \mathsf{U} q_f$ iff there is a finite proof of AVASS whose root is equal to $(q_0,\vec{0})$ and each leaf belongs to $\{q_f\} \times \mathbb{N}^r$. Here, q_f is also understood as a propositional variable.

Before providing a formal definition of $\mathfrak{M}$, we illustrate the construction using a simple example. Assume that $\mathcal{A}$ contains the unary rule $\mathfrak{r}_1 = q_1 \xrightarrow{(-1,+3)} q_0$ and the binary rule $\mathfrak{r}_2 = q_0 \to q_3, q_2$. $\mathfrak{M}$ contains two agents. An inference with $\mathfrak{r}_1$ followed by an inference with $\mathfrak{r}_2$ is simulated using the transitions shown in Fig. 3, where the cost of the action $(\mathfrak{r}_1,\mathfrak{r}_2)$ is precisely equal to $(-1,+3)$ and both $\mathtt{idle}$ and its twin action $\mathtt{idle}'$ have no cost. So, two subsequent rule applications are encoded by one action, which is relevant as the set of control states is made of two disjoint sets of control states that determine strictly whether a unary rule or a fork rule can be applied from them. Note also the presence of a bad state that forbids the choice of the idle action by the first agent, assuming that the objective is to reach the state q_f (with $q_2 = q_f$ presently).

The complete definition of $\mathfrak{M}$ is as follows.

- $Agt \overset{\mathrm{def}}{=} \{1,2\}$. So, the number of agents is independent of the input AVASS.
- A pair of rules $(\mathfrak{r}_1,\mathfrak{r}_2) \in R_1 \times R_2$ is *connected* iff the last control state of $\mathfrak{r}_1$ is equal to the first control state of $\mathfrak{r}_2$. The set of actions Act is equal to the set of connected pairs of rules plus the action $\mathtt{idle}$ and its twin action $\mathtt{idle}'$.
- $S = Q_1 \uplus \{\mathrm{bad}\}$.
- For each control state q in Q_1, $\mathtt{act}(q,1)$ is the set of connected pairs of rules whose unary rule starts from q plus the idle action. So agent 1 can choose a unary rule immediately followed by a fork rule. $\mathtt{act}(q_f,1)$ is restricted to $\{\mathtt{idle}\}$, because no rule starts from q_f in $\mathcal{A}$.
- As far as agent 2 is concerned, for all $q \in Q_1$, $\mathtt{act}(q,2) \overset{\mathrm{def}}{=} \{\mathtt{idle},\mathtt{idle}'\}$. So agent 2 can perform two actions that have no effect on resources, which amounts to simulating the effects of fork rules.
- Only the idle action can be performed from the state bad:

$$\mathtt{act}(\mathrm{bad},1) \overset{\mathrm{def}}{=} \mathtt{act}(\mathrm{bad},2) \overset{\mathrm{def}}{=} \{\mathtt{idle}\}.$$

- The cost of the action $(\mathfrak{r}_1,\mathfrak{r}_2)$ is simply the update vector of the unary rule $\mathfrak{r}_1$. Formally, for all $q \in Q_1$, we have $\mathtt{cost}(q,1,(\mathfrak{r}_1,\mathfrak{r}_2)) \overset{\mathrm{def}}{=} \vec{u}$ when $\mathfrak{r}_1 = (q,\vec{u},q')$ for some q'. Furthermore, $\mathtt{cost}(q,a,\mathtt{idle}) \overset{\mathrm{def}}{=} \mathtt{cost}(\mathrm{bad},a,\mathtt{idle}) \overset{\mathrm{def}}{=} \vec{0}$ for all $a \in \{1,2\}$, and $\mathtt{cost}(q,2,\mathtt{idle}') \overset{\mathrm{def}}{=} \vec{0}$.

- When δ can be defined, we have
  - $\delta(q_f, \mathfrak{f}) \overset{\text{def}}{=} q_f$; $\delta(\text{bad}, \mathfrak{f}) \overset{\text{def}}{=} \text{bad}$ ($\mathfrak{f}$ is the constant map $\texttt{idle}$).
  - Whenever $q \in (Q_1 \setminus \{q_f\})$, $\mathfrak{f}(1) = (\mathfrak{r}_1, \mathfrak{r}_2)$ with $\mathfrak{r}_1$ starting from q and $\mathfrak{r}_2 = (q_{inter}, q', q'')$,

$$\delta(q, \mathfrak{f}) \overset{\text{def}}{=} \begin{cases} q' & \text{if } \mathfrak{f}(2) = \texttt{idle} \\ q'' & \text{otherwise, i.e. } \mathfrak{f}(2) = \texttt{idle}'. \end{cases}$$

  - $\delta(q, \mathfrak{f}) = \text{bad}$ whenever $\mathfrak{f}(1) = \texttt{idle}$ for all $q \in Q_1 \setminus \{q_f\}$.
- There is a unique propositional variable q_f and $Lab(q_f) \overset{\text{def}}{=} \{q_f\}$.

(3) We now establish the correctness of the construction.
Without loss of generality, we can assume $q_0 \neq q_f$.

First, suppose that $\mathfrak{M}, q_0 \models \langle\langle\{1\}^{\vec{0}}\rangle\rangle \top \mathsf{U} q_f$. So, there exists a $\vec{0}$-strategy $F_{\{1\}}$ such that, for all $\lambda = s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \texttt{Comp}(q_0, F_{\{1\}})$, there is an $i \geq 0$ such that $q_i = q_f$ (so by construction of $\mathfrak{M}$, for all $j \geq i$, we have $q_j = q_f$ and bad does not occur in λ). Since $\{1\}$ is a singleton set, we assume below that $F_{\{1\}}$ returns an action for agent 1 (instead of returning a joint action with respect to the single agent 1) and $D_{\{1\}}(q)$ is viewed as an action.

From $\mathfrak{M}$, q_0 and $F_{\{1\}}$, let $(\mathfrak{T}_{F_{\{1\}}}, \mathfrak{R}, \mathfrak{L})$ be the labelled transition system defined in Section 4.1. We have $\mathfrak{T}_{F_{\{1\}}} \subseteq \{1,2\}^*$, $\mathfrak{L} : \mathfrak{T}_{F_{\{1\}}} \to Q_1$ (because bad does not occur in computation from $\texttt{Comp}(q_0, F_{\{1\}})$), and $\mathfrak{R}$ is partial map $\mathfrak{T}_{F_{\{1\}}} \times \mathfrak{T}_{F_{\{1\}}} \to \bigcup_{q \in Q_1} D(q)$. Note the specific structure of $(\mathfrak{T}_{F_{\{1\}}}, \mathfrak{R}, \mathfrak{L})$:

- For all $\mathfrak{w} \in \mathfrak{T}_{F_{\{1\}}}$ such that $\mathfrak{L}(\mathfrak{w}) = q_f$, $\mathfrak{w} \cdot 1$ is the unique successor of $\mathfrak{w}$, $\mathfrak{L}(\mathfrak{w} \cdot 1) = q_f$ and $\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 1)$ is the constant joint action equal to $\texttt{idle}$.
- For all $\mathfrak{w} \in \mathfrak{T}_{F_{\{1\}}}$ such that $\mathfrak{w}$ has exactly two successors $\mathfrak{w} \cdot 1$ and $\mathfrak{w} \cdot 2$, there are $q_{\text{inter}} \in Q$ and $\vec{u} \in \mathbb{Z}^r$ such that

$$\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 1) = ((\overbrace{(\mathfrak{L}(\mathfrak{w}), \vec{u}, q_{\text{inter}})}^{\mathfrak{r}_1}, \overbrace{(q_{\text{inter}}, \mathfrak{L}(\mathfrak{w} \cdot 1), \mathfrak{L}(\mathfrak{w} \cdot 2))}^{\mathfrak{r}_2}), \texttt{idle})$$
$$\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 2) = (((\mathfrak{L}(\mathfrak{w}), \vec{u}, q_{\text{inter}}), (q_{\text{inter}}, \mathfrak{L}(\mathfrak{w} \cdot 1), \mathfrak{L}(\mathfrak{w} \cdot 2))), \texttt{idle}')$$

  and $(\mathfrak{r}_1, \mathfrak{r}_2)$ is connected.

Now, let us build a finite derivation skeleton $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\perp\})$ such that its unique derivation with root labelled by $(q_0, \vec{0})$ is a finite proof with leaves labelled by q_f. The finite tree $\mathfrak{T}$ can be uniquely defined from $\mathfrak{T}_{F_{\{1\}}}$ by using the map $\mathfrak{c} : \{1,2\}^* \to \{1,2\}^*$, where $\mathfrak{c}(\mathfrak{w})$ is obtained from $\mathfrak{w}$ by simultaneously replacing every occurrence of 1 by 11 and every occurrence of 2 by 12. So, for instance, $\mathfrak{c}(\varepsilon) \overset{\text{def}}{=} \varepsilon$ and $\mathfrak{c}(12) \overset{\text{def}}{=} 1112$. We stipulate that $\mathfrak{T}$ is the set of words of the form $\mathfrak{c}(\mathfrak{w})$ or $\mathfrak{c}(\mathfrak{w}) \cdot 1$ where $\mathfrak{w} \in \{1,2\}^*$ and there is $\mathfrak{v} \in \mathfrak{T}_{F_{\{1\}}}$ such that $\mathfrak{w}$ is a (non necessarily strict) prefix of $\mathfrak{v}$, $\mathfrak{L}(\mathfrak{v}) = q_f$ and no strict prefix of $\mathfrak{v}$ is labelled by q_f. The derivation skeleton $\mathcal{D}$ is defined as follows. For all $\mathfrak{w} \in \mathfrak{T}_{F_{\{1\}}}$:

- If $\mathfrak{L}(\mathfrak{w}) \neq q_f$ and $\mathfrak{R}(\mathfrak{w}, \mathfrak{w} \cdot 1) = ((\mathfrak{r}_1, \mathfrak{r}_2), \texttt{idle})$, then $\mathcal{D}(\mathfrak{c}(\mathfrak{w})) \overset{\text{def}}{=} \mathfrak{r}_1$ and $\mathcal{D}(\mathfrak{c}(\mathfrak{w}) \cdot 1) \overset{\text{def}}{=} \mathfrak{r}_2$.
- If $\mathfrak{L}(\mathfrak{w}) = q_f$ and $\mathfrak{c}(\mathfrak{w}) \in \mathfrak{T}$, then $\mathcal{D}(\mathfrak{c}(\mathfrak{w})) = \perp$.

Similarly to Lemma 3, we can show the following properties:

(I) For every computation λ starting at q_0, ending at q_f, visiting q_f only once and respecting $F_{\{1\}}$, there is a maximal branch $\mathfrak{w}$ in $\mathcal{D}$ such that $\texttt{ext}(\lambda, F_{\{1\}}) = \texttt{ext}(\mathfrak{w}, F_{\{1\}})$.
(II) For every maximal branch $\mathfrak{w}$ in $\mathcal{D}$, there is a computation λ starting at q_0, ending at q_f, visiting q_f only once and respecting $F_{\{1\}}$ such that $\texttt{ext}(\mathfrak{w}, F_{\{1\}}) = \texttt{ext}(\lambda, F_{\{1\}})$.

Consequently, the unique derivation based on $\mathcal{D}$ with root labelled by $(q_0, \vec{0})$ is a finite proof with leaves labelled by q_f. Indeed, by construction for all $q \in Q_1$, we have $\texttt{cost}(q, 1, (\mathfrak{r}_1, \mathfrak{r}_2)) = \vec{u}$ when $\mathfrak{r}_1 = (q, \vec{u}, q')$ for some q'.

For the converse direction, let us assume the existence of a finite proof $\hat{\mathcal{D}}$ based on the derivation skeleton $\mathcal{D} : \mathfrak{T} \to (R_1 \cup R_2 \cup \{\perp\})$ such that $\hat{\mathcal{D}}(\varepsilon) = (q_0, \vec{0})$ and each leaf is labelled by a pair in $\{q_f\} \times \mathbb{N}^r$. Let us define a strategy $F_{\{1\}}$. First, we require the following properties:

- $F_{\{1\}}(q_0) \overset{\text{def}}{=} (\mathcal{D}(\varepsilon), \mathcal{D}(1))$. Since $q_0 \neq q_f$, we know that $1 \in \mathfrak{T}$.
- For all the finite computations λ ending at the state q_f (we have $Q_1 \subseteq S$ and $q_f \in Q_1$), $F_{\{1\}}(\lambda) \overset{\text{def}}{=} \texttt{idle}$.

Let $\lambda = q_0 \xrightarrow{\mathfrak{g}_0} q_1 \xrightarrow{\mathfrak{g}_1} q_2 \cdots \xrightarrow{\mathfrak{g}_{n-1}} q_n$ be a finite computation respecting (so far) $F_{\{1\}}$ and $q_n \neq q_f$ (since this case is already treated above). Below we define $F_{\{1\}}(\lambda)$.

First, we assume that bad does not occur in λ. Each joint action $\mathfrak{g}_j$ can be written $((\mathfrak{r}_1^j, \mathfrak{r}_2^j), \mathfrak{b}(k^j))$ with $k^j \in \{1, 2\}$ and $\mathfrak{b} : \{1, 2\} \to \{\texttt{idle}, \texttt{idle}'\}$ with $\mathfrak{b}(1) \stackrel{\text{def}}{=} \texttt{idle}$ and $\mathfrak{b}(2) \stackrel{\text{def}}{=} \texttt{idle}'$. The derivation skeleton $\mathcal{D}$ verifies the properties below:

- $\mathcal{D}(\varepsilon) = \mathfrak{r}_1^0$ with $\mathfrak{r}_1^0 = (q_0, \vec{u}_0, q^0_{\text{inter}})$.
- $\mathcal{D}(1) = \mathfrak{r}_2^0$ with $\mathfrak{r}_2^0 = (q^0_{\text{inter}}, q_1^0, q_2^0)$ and $q^0_{k^0} = q_1$.
- $\mathcal{D}(1k^0) = \mathfrak{r}_1^1$ with $\mathfrak{r}_1^1 = (q^0_{k^0}, \vec{u}_1, q^1_{\text{inter}})$.
- $\mathcal{D}(1k^0 1) = \mathfrak{r}_2^1$ with $\mathfrak{r}_2^1 = (q^1_{\text{inter}}, q_1^1, q_2^1)$ and $q^1_{k^1} = q_2$.
- ...
- $\mathcal{D}(1k^0 1 \cdots k^j) = \mathfrak{r}_1^{j+1}$ with $\mathfrak{r}_1^{j+1} = (q_{k^j}, \vec{u}_{j+1}, q^{j+1}_{\text{inter}})$.
- $\mathcal{D}(1k^0 1 \cdots k^j 1) = \mathfrak{r}_2^{j+1}$ with $\mathfrak{r}_2^{j+1} = (q^{j+1}_{\text{inter}}, q_1^{j+1}, q_2^{j+1})$ and $q^{j+1}_{k^{j+1}} = q_{j+2}$.
- ...
- $\mathcal{D}(1k^0 1 \cdots k^{n-2} 1) = \mathfrak{r}_2^{n-1}$ with $\mathfrak{r}_2^{n-1} = (q^{n-1}_{\text{inter}}, q_1^{n-1}, q_2^{n-1})$ and $q^{n-1}_{k^{n-1}} = q_n$.

Since q_n is different from q_f, $1k^0 1 \cdots k^{n-2} 1 k^{n-1}$ and $1k^0 1 \cdots k^{n-2} 1 k^{n-1} 1$ exist. We stipulate $F_{\{1\}}(\lambda) \stackrel{\text{def}}{=} (\mathcal{D}(1k^0 1 \cdots k^{n-2} 1 k^{n-1}), \mathcal{D}(1k^0 1 \cdots k^{n-2} 1 k^{n-1} 1))$. Consequently, any extension $q_0 \xrightarrow{\mathfrak{g}_0} q_1 \xrightarrow{\mathfrak{g}_1} q_2 \cdots \xrightarrow{\mathfrak{g}_{n-1}} q_n \xrightarrow{\mathfrak{g}_n} q_{n+1}$ respecting (again so far) $F_{\{1\}}$ verifies the above correspondence with $\mathcal{D}$ and the state bad cannot be visited. So, the strategy $F_{\{1\}}$ can be defined by using the approach above (more formally, an induction hypothesis should be stated and we should prove that after each step, the property is preserved). One can also check that $F_{\{1\}}$ is $\vec{0}$-consistent w.r.t q_0, and, for all $\lambda = s_0 \xrightarrow{\mathfrak{g}_0} s_1 \ldots \in \texttt{Comp}(q_0, F_{\{1\}})$, there is $i \geq 0$ such that $q_i = q_f$. The $\vec{0}$-consistency is due to the fact that $\hat{\mathcal{D}}$ is a proof and the reachability condition is a consequence of the fact that every leaf of $\hat{\mathcal{D}}$ is labelled by q_f because of the correspondences between computations respecting $F_{\{1\}}$ and nodes in $\mathfrak{T}$. $\square$

The hardness proof above would also work if the proponent restriction is not satisfied, or if no distinguished idle action is assumed in the game structures, or if $\texttt{act}$ may return an empty set of actions. Indeed, the construction of $\mathfrak{M}$ in the proof of Theorem 3 assumes the proponent restriction condition but this condition is actually useless as all the actions for the agent 2 have zero cost. Similarly, the construction of $\mathfrak{M}$ takes into account the existence of a special action $\texttt{idle}$ (and this entails a few complications) but more generally, $\texttt{idle}$ can be also viewed as a non-distinguished action and therefore the hardness proof is not sensitive to the existence of a distinguished idle action. Last but not least, allowing that $\texttt{act}$ may return an empty set of actions is compatible with the current construction of $\mathfrak{M}$ (this extra freedom is therefore not used to build $\mathfrak{M}$). By construction of $\mathfrak{M}$, it is also worth observing that one propositional variable and two agents are sufficient to get 2EXPTIME-hardness.

In the corollary below, we use [33, Theorem 3.4] and [17, Theorem 3.1], which show that for a bounded number of resources, the state reachability and the non-termination problems for AVASS can be solved in EXPTIME. When $r \geq 4$, the state reachability problem for AVASS is EXPTIME-hard [33], which leads to the result below.

Corollary 1. *For any fixed $r \geq 1$, the model checking problem for RB±ATL restricted to at most r resources is in* EXPTIME. *For $r \geq 4$, the problem is* EXPTIME*-hard.*

Moreover, if r is fixed but greater than two, then the model checking problem for RB±ATL restricted to at most r resources is PSPACE-hard, since the state reachability problem for VASS of dimension two is PSPACE-complete [9]. When $r = 1$, the model checking problem for RB±ATL is NP-hard since the state reachability for VASS of dimension one is NP-complete [27]. (Note that the NP-completeness result does not apply because the model checking problem for RB±ATL involves not just the reachability problem but also the non-termination problem.)

We have seen that the model checking problem for RB±ATL restricted to two agents is 2EXPTIME-hard; below we show that the restriction to a single agent is only EXPSPACE-complete.

Theorem 4. *The model checking problem for RB±ATL restricted to a single agent is* EXPSPACE*-complete.*

Proof. In order to show the EXPSPACE upper bound, we sketch how to solve the model checking problem for RB±ATL restricted to a single agent, by solving instances of the model checking problem for LTL on VASS or instances of the model checking problem for CTL, known to be EXPSPACE-complete (see e.g. [28]) and P-complete (see e.g. [42]) respectively. The labelling algorithm has exactly the same form as the algorithm for full RB±ATL. The size of the instances of the problems is linear in the size of the input resource-bounded concurrent game structures, and the number of calls is also linear in the size of the input formulae. This leads to the EXPSPACE upper bound.

The following two properties are essential for the proof. Given a path formula Φ of the form $\mathsf{X}p$, $\mathsf{G}p$ or $p_1 \mathsf{U} p_2$, one can show that $\mathfrak{M}, s \models \langle\langle \emptyset^{\vec{b}} \rangle\rangle \Phi$ iff $\mathfrak{M}', s \models \mathsf{A}\Phi$ in CTL, where $\mathfrak{M}'$ is obtained from $\mathfrak{M}$ by removing the costs and actions from the transitions. Note that the empty coalition $\emptyset$ allows us to quantify over all computations, and therefore the value of the bound $\vec{b}$ is irrelevant. Similarly, one can show that $\mathfrak{M}, s \models \langle\langle\{1\}^{\vec{b}}\rangle\rangle\Phi$ iff there is an infinite run from the initial configuration $(s, \vec{b})$ in the VASS $\mathcal{V}$ that satisfies the LTL formula Φ, where $\mathcal{V}$ is obtained from $\mathfrak{M}$ by removing the actions while keeping the costs in $\mathbb{Z}^r$ on the transitions. If $\vec{b}$ has components with the value ω, then we reduce the dimension in $\vec{b}$ and in $\mathcal{V}$ so that only the components with finite values in $\vec{b}$ remain.

In order to get the EXPSPACE lower bound, we reduce the state reachability problem for VASS to the model checking problem for RB±ATL restricted to a single agent, and use the EXPSPACE-hardness established in [37]. Let $\mathcal{V} = (Q, r, R)$ be a VASS and q_0, q_f be locations. One can show that there is a run from $(q_0, \vec{0})$ to some configuration of the form $(q_f, \vec{x})$ for some $\vec{x} \in \mathbb{N}^r$ iff $\mathfrak{M}, q_0 \models \langle\langle\{1\}^{\vec{0}}\rangle\rangle \top \mathsf{U} p$, where $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, Lab)$ is defined from $\mathcal{V}$ as follows (for all $q, q' \in Q$):

- $Agt = \{1\}$, $S = Q \uplus \{\text{bad}\}$ and $Act = R \uplus \{\mathtt{idle}\}$.
- Lab is defined so that p holds true exactly on q_f and $\mathtt{cost}(1, q \xrightarrow{\vec{u}} q') = \vec{u}$.
- $\mathtt{act}(q, 1) = \{\mathtt{idle}\} \uplus \{q \xrightarrow{\vec{u}} q' \mid q \xrightarrow{\vec{u}} q' \in R\}$, $\mathtt{act}(\text{bad}, 1) = \{\mathtt{idle}\}$.
- Finally, $\delta(q, q \xrightarrow{\vec{u}} q') = q'$, $\delta(q, \mathtt{idle}) = \text{bad}$ and $\delta(\text{bad}, \mathtt{idle}) = \text{bad}$. □

5. Resource-bounded temporal logics RBTL and RBTL*

In this section, we present the logic RBTL introduced in [12] and its extension RBTL* and we characterise the computational complexity of the model-checking problem. The proof can be seen as a simpler version of the proof for RB±ATL and a more complex version of the standard proof for CTL*.

5.1. The logic RBTL* and its variants

The models of the logic RBTL* are structures of the form (Q, r, R, Lab) where (Q, r, R) is a VASS and Lab is a labelling built on elements of Q understood as propositional variables, so that $Lab(q) = \{q\}$ (see e.g., [12, Section 3]). "RBTL" stands for 'resource-bounded temporal logic' and the logic RBTL defined below (a fragment of RBTL*) has been introduced in [12]. For consistency with standard terminology, an infinite proof in (Q, r, R) is called a *path* or *run*, and is represented by $\lambda = (q_0, \vec{v}_0) \to (q_1, \vec{v}_1) \ldots$. We write $\lambda(i)$ to denote the ith configuration $(q_i, \vec{v}_i)$, and $\lambda[+i, +\infty)$ to denote the suffix of λ starting from $(q_i, \vec{v}_i)$.

The *state formulae* ϕ and the *path formulae* Φ of RBTL* are defined mutually recursively by the following grammar (relative to a set of locations Q and number of resources r, which is not a significant restriction since we are only interested in model checking)

$$\begin{aligned} \phi &::= q \mid \neg\phi \mid (\phi \wedge \phi) \mid \langle \vec{b} \rangle\, \Phi \\ \Phi &::= \phi \mid \neg\Phi \mid (\Phi \wedge \Phi) \mid \mathsf{X}\,\Phi \mid (\Phi \mathsf{U} \Phi) \mid \mathsf{G}\Phi, \end{aligned}$$

where $q \in Q$. Syntactically, every state formula is also a path formula according to this grammar, reflecting the fact that a path uniquely identifies a location in which a formula is interpreted (its starting location).

In presenting the semantics of RBTL*, we make an explicit distinction between state formulae and path formulae. The two satisfaction relations $\models_{\mathfrak{s}}$ and $\models_{\mathfrak{p}}$ are defined as follows (standard clauses for the Boolean connectives are again omitted):

$$\begin{array}{lcl} \mathfrak{M}, q \models_{\mathfrak{s}} q' & \overset{\text{def}}{\Leftrightarrow} & q' = q \\ \mathfrak{M}, q \models_{\mathfrak{s}} \langle \vec{b} \rangle \Phi & \overset{\text{def}}{\Leftrightarrow} & \text{there is an infinite run } \lambda \text{ starting at } (q, \vec{b}) \\ & & \text{such that } \mathfrak{M}, \lambda \models_{\mathfrak{p}} \Phi \\ \mathfrak{M}, \lambda \models_{\mathfrak{p}} \phi & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda(0) \models_{\mathfrak{s}} \phi \text{ for state formulae } \phi \\ \mathfrak{M}, \lambda \models_{\mathfrak{p}} \mathsf{X}\Phi & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda[1, +\infty) \models_{\mathfrak{p}} \Phi \\ \mathfrak{M}, \lambda \models_{\mathfrak{p}} \Phi \mathsf{U} \Psi & \overset{\text{def}}{\Leftrightarrow} & \text{there is } i \geq 0 \text{ such that } \mathfrak{M}, \lambda[i, +\infty) \models_{\mathfrak{p}} \Psi \text{ and} \\ & & \text{for every } j \in [0, i-1], \text{ we have } \mathfrak{M}, \lambda[j, +\infty) \models_{\mathfrak{p}} \Phi. \end{array}$$

As usual, we write $[\vec{b}]\phi$ to denote the formula $\neg\langle\vec{b}\rangle\neg\phi$, and therefore $\mathfrak{M}, q \models_{\mathfrak{s}} [\vec{b}]\Phi$ iff for all the infinite runs λ starting at $(q, \vec{b})$, we have $\mathfrak{M}, \lambda \models_{\mathfrak{p}} \Phi$.

The model checking problem for RBTL* is defined as follows:

Input: A model $\mathfrak{M} = (Q, r, R, Lab)$, a control state q and a state formula ϕ.
Question: $\mathfrak{M}, q \models_{\mathfrak{s}} \phi$?

As CTL is a syntactic fragment of CTL*, RBTL is defined as the syntactic fragment of RBTL* in which any subformula with an outermost connective in $\{\mathsf{U}, \mathsf{X}, \mathsf{G}\}$ is immediately preceded by a modality of the form either $\langle\vec{b}\rangle$ or $[\vec{b}]$. Observe that the model checking problem for RBTL is already EXPSPACE-hard, since the state reachability problem for VASS can be reduced to a question of the form $\mathfrak{M}, q_0 \models \langle \vec{0} \rangle\ q_f$. We consider the computational complexity of the model checking problems for RBTL* and RBTL in Section 5.2.

It is worth noting that the definition of RBTL above is taken from [12] and other variants would be possible, for instance to interpret the formulae on other classes of counter machines (arbitrary transition systems are not possible as $\langle \vec{b} \rangle$ provides initial conditions based on counter values). As for CTL*, path quantifiers quantify over all possible paths but path formulae are interpreted on a single path, which explains why the notion of runs in VASS is essential to define the semantics of RBTL*.

5.2. *On the complexity of the model-checking problem for RBTL**

In this section, we study the model checking problem for resource-bounded logics in which the path formulae are arbitrary, i.e., they can be any LTL-like formulae rather than being restricted to path formulae of the form $\mathsf{G}\psi$, $\mathsf{X}\psi$ and $\psi_1\ \mathsf{U}\ \psi_2$ as in RB±ATL. We have already seen that the model checking problem for RBTL is EXPSPACE-hard (see Section 5.1) and therefore the lower bound also applies to RBTL*. Below, we show that the model checking problem for RBTL* is not only decidable (a new result) but also in EXPSPACE. The arguments for establishing the EXPSPACE upper bound for RBTL and RBTL* are identical, and the EXPSPACE lower bound for the model checking problem for RBTL can be matched with the upper bound for RBTL*.

Theorem 5. *The model checking problem for RBTL* is in* EXPSPACE.

In [12], the problem for RBTL is shown to be decidable by reduction to the reachability problem for VASS. However the best known upper bound for the reachability problem is quite high, see e.g. [36]. Hence, the EXPSPACE upper bound is a substantial improvement. Moreover, the decidability of the model checking problem for RBTL* was left open in [12].

The proof of Theorem 5 is inspired from the proof of the PSPACE upper bound for CTL* model checking based on LTL model checking. The main difference rests on the fact that, herein, the subroutine involving LTL is performed for VASS instead of for finite-state transition systems.

Proof. (sketch) The algorithm to obtain the EXPSPACE upper bound first computes the states in which subformulae hold before dealing with larger formulae. The algorithm is a renaming algorithm. However, there is a caveat: when dealing with subformulae of the form $\langle \vec{b} \rangle \Psi$ where Ψ is an LTL formula, we are entitled to use the model checking algorithm for LTL formulae on VASS that is in EXPSPACE [28] (having ω in one component amounts to ignoring that position). However, in order to systematically consider such subformulae $\langle \vec{b} \rangle \Psi$ when the outermost connective is a path quantifier, we need to perform renamings on-the-fly.

Let us provide a simple example with the formula

$$\phi = \langle \vec{b}_0 \rangle\ \mathsf{GF}\ \langle \vec{b}_1 \rangle\ q\mathsf{U}q'$$

and an arbitrary model $\mathfrak{M}$. Let us consider some innermost state formula prefixed by a path quantifier, say $\langle \vec{b}_1 \rangle (q\mathsf{U}q')$ (actually here there is only one such a subformula). With the help of a decision procedure for solving the LTL model checking problem on VASS, we determine for which control states q'' we have $\mathfrak{M}, q'' \models_{\mathfrak{s}} \langle \vec{b}_1 \rangle (q\mathsf{U}q')$. Say, we obtain the set $\{q_1, \ldots, q_\alpha\}$. Now, in ϕ, we replace $\langle \vec{b}_1 \rangle (q\mathsf{U}q)$ by $q_1 \vee \cdots \vee q_\alpha$, and we get $\phi_1 = \langle \vec{b}_0 \rangle\ \mathsf{GF}\ (q_1 \vee \cdots \vee q_\alpha)$. So, we have performed a renaming step by replacing a subformula by a disjunction of propositional variables. This process can be repeated until there are no more path quantifiers. To do this, we substitute some innermost state formula prefixed by a path quantifier in ϕ_1, say $\langle \vec{b}_0 \rangle \mathsf{GF}\ (q_1 \vee \cdots \vee q_\alpha)$, by a new disjunction of locations (possibly empty) with the help of a decision procedure for solving the LTL model checking problem on VASS. We obtain the manageable formula

$$\phi_2 = q'_1 \vee \cdots \vee q'_\beta.$$

Since ϕ_2 is a propositional formula, we are done with the renaming process and it is easy to show that for every control state q'', we have $\mathfrak{M}, q'' \models_{\mathfrak{s}} \phi$ iff $\mathfrak{M}, q'' \models_{\mathfrak{s}} q'_1 \vee \cdots \vee q'_\beta$. The above example can be easily generalised to any state formula. Note that the number of renamings is bounded by the size of the input formula, and at each step a subroutine is invoked at most $\text{card}(Q)$ times and requires EXPSPACE, whence we obtain the EXPSPACE upper bound. □

Corollary 2. *For any fixed $r \geq 1$, the model checking problem for RBTL* restricted to at most r resources is in* PSPACE.

The PSPACE upper bound is a consequence of [28]. The model checking problem for LTL on VASS is in PSPACE when the number of counters is bounded [28, Theorem 4.1], and the renaming algorithm briefly described in the proof of Theorem 5 makes only a linear number of calls to the model checking problem for LTL on VASS and the number of counters is preserved when such calls are performed. If r is fixed but greater than two, then the model checking problem for RBTL* restricted to at most r resources is PSPACE-hard, since the state reachability problem for VASS of dimension two is PSPACE-complete [9].

When $r = 1$, the model checking problem for RBTL* is NP-hard, since the state reachability problem for VASS of dimension one is NP-complete [27].

6. The logic RB±ATL* and its parameterised variant

We have seen that the model checking problem for RB±ATL is 2EXPTIME-complete. Below, we show that the model checking problem for RB±ATL* is also decidable. The arguments for establishing the respective decidability of RB±ATL and RB±ATL* both rest on the decidability of decision problems for alternating VASS. However for the model checking problem for RB±ATL* we need to invoke the decidability of parity games on alternating VASS, which is stronger than the decidability of the state reachability and non-termination problems for AVASS. This more complex reduction, which uses ingredients such as the standard equivalence of expressive power of Büchi automata and deterministic parity automata on ω-words, is nevertheless rewarding, as it allows us to synthetise concrete values for resource parameters, something which has heretofore not been possible for resource-bounded logics.

Below, we introduce RB±ATL*, an extension of RB±ATL in which the path formulae are unconstrained, i.e. they can be any LTL-like formula. Although RB±ATL* is a new logic, its definition follows a standard schema for branching-time temporal logics.

6.1. Definition

Given a set of agents $Agt = \{a_1, \ldots, a_k\}$ and $r \geq 1$, we write RB±ATL*(Agt, r) to denote the resource-bounded logic with k agents and r resources whose models are resource-bounded concurrent game structures with the same parameters. Formulae of RB±ATL*(Agt, r) are defined according to the grammar below (as in CTL* or for RBTL*, we distinguish between state formulae ϕ and path formulae Φ)

$$\phi \ ::= \ p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A^{\vec{b}} \rangle\rangle \ \Phi$$

$$\Phi \ ::= \ \phi \mid \neg\Phi \mid (\Phi \wedge \Phi) \mid \mathsf{X}\ \Phi \mid (\Phi \mathsf{U} \Phi) \mid \mathsf{G}\Phi,$$

where $p \in \mathrm{PROP}$, $A \subseteq Agt$ and $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$. The set of state formulae ϕ for RB±ATL*(Agt, r) extends the set of formulae for RB±ATL(Agt, r).[3] In presenting the semantics for RB±ATL*, we make an explicit distinction between state formulae and path formulae. The two satisfaction relations $\models_{\mathfrak{s}}$ and $\models_{\mathrm{p}}$ are defined as follows.

$$
\begin{array}{lcl}
\mathfrak{M}, s \models_{\mathfrak{s}} p & \overset{\text{def}}{\Leftrightarrow} & s \in Lab(p) \\
\mathfrak{M}, s \models_{\mathfrak{s}} \langle\langle A^{\vec{b}} \rangle\rangle \ \Phi & \overset{\text{def}}{\Leftrightarrow} & \text{there is a } \vec{b}\text{-strategy } F_A \text{ w.r.t. } s \text{ such that} \\
 & & \text{for all } \lambda = s_0 \xrightarrow{\mathfrak{f}_0} s_1 \ldots \in \mathrm{Comp}(s, F_A), \text{ we have } \mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi \\
\mathfrak{M}, \lambda \models_{\mathrm{p}} \phi & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda(0) \models_{\mathfrak{s}} \phi \text{ for state formulae } \phi \\
\mathfrak{M}, \lambda \models_{\mathrm{p}} \neg\Phi & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda \not\models_{\mathrm{p}} \Phi \\
\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi \wedge \Phi' & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi \text{ and } \mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi' \\
\mathfrak{M}, \lambda \models_{\mathrm{p}} \mathsf{X}\Phi & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda[1, +\infty) \models_{\mathrm{p}} \Phi \\
\mathfrak{M}, \lambda \models_{\mathrm{p}} \mathsf{G}\Phi & \overset{\text{def}}{\Leftrightarrow} & \mathfrak{M}, \lambda[i, +\infty) \models_{\mathrm{p}} \Phi \text{ for all } i < |\lambda| \\
\mathfrak{M}, \lambda \models_{\mathrm{p}} \Phi \mathsf{U} \Psi & \overset{\text{def}}{\Leftrightarrow} & \text{there is } i < |\lambda| \text{ such that } \mathfrak{M}, \lambda[i, +\infty) \models_{\mathrm{p}} \Psi \text{ and} \\
 & & \text{for every } j \in [0, i-1], \text{ we have } \mathfrak{M}, \lambda[j, +\infty) \models_{\mathrm{p}} \Phi.
\end{array}
$$

Again, all the maximal computations are infinite, i.e., the index i in the clauses for G or U can take any value in $\mathbb{N}$. The *model checking problem for RB±ATL** is defined as follows:

Input: $k, r \geq 1$ (in unary), a state formula ϕ in RB±ATL*$([1, k], r)$, a finite model $\mathfrak{M}$ and a state s,
Question: $\mathfrak{M}, s \models_{\mathfrak{s}} \phi$?

Below, we show that the model checking problem for RB±ATL* is decidable by reduction to the parity game problem for single-sided VASS [1, Corollary 2]. The latter problem will play a role similar to LTL model checking in CTL* model checking, see e.g. [42,20]. In addition, [1, Theorem 4] allows us to synthetise resource bounds. We begin by defining a variant of the problem.

[3] G can be encoded using U and ¬, but we retain it to emphasize that we are dealing with an extension of RB±ATL(Agt, r).

6.2. A parameterised variant

We first introduce a *parameterised version* of RB±ATL*, denoted by ParRB±ATL*. The formulae of ParRB±ATL* are the same as those of RB±ATL*, except that the concrete values $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ that decorate strategy modalities are replaced by tuples of variables taken from the set $\mathtt{VAR} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$, for example:

$$\langle\langle\{1\}^{(\mathtt{x}_1,\mathtt{x}_2)}\rangle\rangle \top \mathsf{U} q_f \wedge \langle\langle\{2\}^{(\mathtt{x}_2,\mathtt{x}_3)}\rangle\rangle \top \mathsf{U} q'_f.$$

Given a parameterised (state or path) formula ϕ with variables $\mathtt{x}_1, \ldots, \mathtt{x}_n$ and a map $\mathfrak{v} : \{\mathtt{x}_1, \ldots, \mathtt{x}_n\} \to (\mathbb{N} \cup \{\omega\})$ (sometimes called a *concretisation*), we write $\mathfrak{v}(\phi)$ to denote the formula in RB±ATL* obtained from ϕ by replacing each occurrence of a variable $\mathtt{x}$ by $\mathfrak{v}(\mathtt{x})$. The *parameterised model checking problem for ParRB±ATL** is defined as follows:

Input: $k, r \geq 1$ (in unary), a parameterised state formula ϕ in ParRB±ATL*$([1,k], r)$, a finite RB±ATL*$([1,k], r)$ model $\mathfrak{M}$ and a state s,
Question: Compute the set of maps $\mathfrak{v}$ such that $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\phi)$.

By 'compute the set of maps', we mean being able to characterise the set of maps $\mathfrak{v}$ such that $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\phi)$, by using a symbolic representation with nice computational properties. More precisely, we shall consider *constrained formulae* following the grammar below:

$$\psi ::= \mathtt{x} \geq c \mid \mathtt{x} = \omega \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi,$$

where $\mathtt{x} \in \mathtt{VAR}$ and $c \in \mathbb{N}$. Given such constraints, it is easy to check non-emptiness or to check the satisfaction of $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\phi)$ for a specific map $\mathfrak{v}$. To synthetise such parameters we use a remarkable result from single-sided VASS: the Pareto frontier for any parity game on single-sided VASS is computable [1, Theorem 4].

6.3. Parity acceptance condition

Below, we consider AVASS with a finite set of fork rules included in $\bigcup_{\beta \geq 2} Q^{\beta}$, and where the proofs are trees with nodes labelled by elements in $Q \times (\mathbb{N} \cup \{\omega\})^r$. Given an AVASS $\mathcal{A} = (Q, r, R_1, R_2)$, a *colouring* $\mathtt{col}$ (a.k.a. a *priority function*) is defined as a map $Q \to [0, \mathfrak{p} - 1]$ for some $\mathfrak{p} \geq 1$ (number of *priorities*). The *parity game problem* for AVASS is defined as follows:

Input: An alternating VASS $\mathcal{A}$, a control state q_0, $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ and $\mathtt{col} : Q \to [0, \mathfrak{p} - 1]$.
Question: Is there a proof whose root is equal to $(q_0, \vec{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even (the colour of each configuration is induced by $\mathtt{col}$) ?

Proposition 4. *[1, Corollary 2] The parity game problem for alternating VASS is decidable.*

To be precise, [1, Corollary 2] states the result for single-sided VASS. A single-sided VASS can be viewed as an alternating VASS where the set Q of control states can be partitioned into $Q = Q_1 \uplus Q_2$, unary rules start from states in Q_1, fork rules start from states in Q_2 and there is at most one fork rule starting from the same control state (necessarily, it belongs to Q_2). The construction of two disjoint sets Q_1 and Q_2 with alternation of unary rules and fork rules can be done as in part (1) in the proof of Theorem 3. However, the colour of the new control states is equal to zero so that it has no influence on the acceptance parity condition. In order to guarantee the uniqueness of fork rules starting from a given control state, it is sufficient to replace any unary rule $q \xrightarrow{\vec{u}} q'$ and fork rule $\mathfrak{r} = (q', q_1, q_2)$ by the unary rule $q \xrightarrow{\vec{u}} (q', \mathfrak{r})$ and the fork rule $((q', \mathfrak{r}), q_1, q_2)$. The colour of $(q', \mathfrak{r})$ is the colour of q'. With such a transformation, decidability for single-sided VASS can be lifted to alternating VASS (this also implies a reduction for the computation of the Pareto frontier below).

It is not difficult to show that the state reachability and non-termination problems for AVASS are subproblems of the parity game problem, and therefore their decidability also follows from [1]. The decidability result for the parity game problem for AVASS is strengthened in [1] with the computation of Pareto frontiers, as briefly explained below. Given $\mathcal{A}$, q_0 and $\mathtt{col} : Q \to [0, \mathfrak{p} - 1]$, the set of tuples $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ for which there is a proof such that the root is equal to $(q_0, \vec{b})$, all the maximal branches are infinite, and for each infinite branch the maximal colour that appears infinitely often is even, is upward closed and computable. This means that it can be represented effectively by a Boolean combination of atomic constraints of the form $\mathtt{x}_i \geq c$ where $i \in [1, r]$ and $c \in \mathbb{N}$, and $\mathtt{x}_i = \omega$. Since the set is upward closed, by Dickson's Lemma, it has a finite set of minimal elements (with respect to the well-quasi-ordering $\preceq$ slightly extended to accommodate the addition of the value ω), allowing the symbolic representation in terms of atomic constraints of the form $\mathtt{x} \geq c$ to be easily defined. The *Pareto frontier* of $\mathcal{A}$, q_0 and $\mathtt{col} : Q \to [0, \mathfrak{p} - 1]$ is defined as the set of minimal elements in $(\mathbb{N} \cup \{\omega\})^r$ for which there is a positive solution to the parity game problem.

Proposition 5. *[1, Theorem 4] The Pareto frontier for any parity game on single-sided VASS is computable.*

6.4. A synchronised product

Before defining the reduction from the model checking problem for RB±ATL* to the parity game problem, we need to introduce a few more definitions, and in particular a notion of synchronisation that will be useful in the sequel.

Let $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, Lab)$ be a resource-bounded concurrent game structure. Given the propositional variables $p_1, \ldots, p_n$, we write $\Sigma_n \overset{\text{def}}{=} \mathcal{P}(\{p_1, \ldots, p_n\})$ to denote the finite alphabet and $Lab_n(s') \overset{\text{def}}{=} \{p_i \mid i \in [1, n],\ s' \in Lab(p_i)\}$ for all $s' \in S$. So, by definition $Lab_n(s') \in \Sigma_n$.

Let $\mathcal{A}_{\mathfrak{M},A,s^\star} = (Q, r, R_1, R_2)$ be the AVASS defined from $\mathfrak{M}$, A and $s^\star$ (see Section 4.1), and $\mathbb{A} = (Q', q'_0, \Delta : Q' \times \Sigma_n \to Q', \mathtt{col} : Q' \to [0, \mathfrak{p} - 1])$ be a deterministic parity automaton over the alphabet Σ_n. We recall that in $\mathbb{A}$,

- Q' is a finite set of states,
- q'_0 is the initial state,
- $\Delta : Q' \times \Sigma_n \to Q'$ is the transition function and,
- $\mathtt{col} : Q' \to [0, \mathfrak{p} - 1]$ is the priority function that induces the acceptance condition and $\mathfrak{p}$ is the number of priorities.

An ω-word $\sigma = \mathtt{a}_0\mathtt{a}_1\mathtt{a}_1 \cdots \in \Sigma_n^\omega$ is accepted by $\mathbb{A}$ (also written $\sigma \in \mathrm{L}(\mathbb{A})$) iff there is a run $q'_0 \xrightarrow{\mathtt{a}_0} q'_1 \xrightarrow{\mathtt{a}_1} q'_2 \cdots$ for which for all $i \geq 0$, we have $\Delta(q'_i, \mathtt{a}_i) = q'_{i+1}$ and

$$\max\{c \in [0, \mathfrak{p} - 1] \mid \{i \in \mathbb{N} \mid \mathtt{col}(q'_i) = c\} \text{ is infinite}\}$$

is even.

The principle of the *synchronised product* $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$ defined below is the following. Any (infinite) branch of a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ contains control states of the form $s^\star$, $(s', \mathfrak{f})$ or $(\mathfrak{g}, s')$ where $s^\star$ is a distinguished state of $\mathfrak{M}$, s' is any state, $\mathfrak{f} \in D_A(s')$ and $\mathfrak{g}$ is joint action in $D(s'')$ with $\delta(s'', \mathfrak{g}) = s'$. By construction, $(s', \mathfrak{f})$ is preceded by a state of the form either $(\mathfrak{g}, s')$ or s' (if $s' = s^\star$). So an infinite branch of the form

$$(s_0, \vec{u}_0)\ ((s_0, \mathfrak{f}_0), \vec{u}_1)\ ((\mathfrak{g}_1, s_1), \vec{u}_1)\ ((s_1, \mathfrak{f}_1), \vec{u}_2)\ ((\mathfrak{g}_2, s_2), \vec{u}_2) \cdots$$

leads to the ω-word

$$Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2) \cdots$$

that corresponds to a unique run in $\mathbb{A}$ (because $\mathbb{A}$ is deterministic and complete and $Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2) \cdots$ is in Σ_n^ω).[4] The control states of $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$ are pairs in $Q \times Q'$ and the second components are therefore control states in Q' for the run on $Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2) \cdots$.

Let us define the AVASS $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A} \overset{\text{def}}{=} (Q'', r, R'_1, R'_2)$ such that:

- $Q'' \overset{\text{def}}{=} Q \times Q'$.
- For each unary rule $s^\star \xrightarrow{\vec{u}} (s^\star, \mathfrak{f}) \in R_1$, in R'_1 we have the unary rule $(s^\star, q'_0) \xrightarrow{\vec{u}} ((s^\star, \mathfrak{f}), q'_0)$.
- For each unary rule $(\mathfrak{g}, s') \xrightarrow{\vec{u}} (s', \mathfrak{f}) \in R_1$ and each $q \in Q'$, in R'_1 we have the unary rule $((\mathfrak{g}, s'), q) \xrightarrow{\vec{u}} ((s', \mathfrak{f}), q)$. So, firing a unary rule from $\mathcal{A}_{\mathfrak{M},A,s^\star}$ does not change the second component.
- For each fork rule $((s', \mathfrak{f}), (\mathfrak{g}_1, s_1), \ldots, (\mathfrak{g}_\alpha, s_\alpha)) \in R_2$, and for each $q \in Q'$, in R'_2 we have the fork rule
$$(((s', \mathfrak{f}), q), ((\mathfrak{g}_1, s_1), q'), \ldots, ((\mathfrak{g}_\alpha, s_\alpha), q')),$$
with $q' = \Delta(q, Lab_n(s'))$. So, firing a fork rule from $\mathcal{A}_{\mathfrak{M},A,s^\star}$ does change the second component in a unique way depending on q and on the letter $Lab_n(s')$. Again, there is a unique fork rule starting from the control state $((s', \mathfrak{f}), q)$.

Let us define the colouring $\mathtt{col}' : Q'' \to [0, \mathfrak{p} - 1]$ such that for all $(q, q') \in Q''$, we have $\mathtt{col}'((q, q')) \overset{\text{def}}{=} \mathtt{col}(q')$. This is the most natural way to inherit colours from $\mathbb{A}$ to $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$.

Lemma 10. *Let $(s^\star, \vec{b}) \in Q \times (\mathbb{N} \cup \{\omega\})^r$. The statements below are equivalent.*

(I) *$\mathcal{A}_{\mathfrak{M},A,s^\star}$ has a proof whose root is equal to $(s^\star, \vec{b})$, all the maximal branches are infinite and the Lab_n-projection of each infinite branch belongs to $\mathrm{L}(\mathbb{A})$.*

(II) *$\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$ has a proof whose root is equal to $((s^\star, q'_0), \vec{b})$, all the maximal branches are infinite and for all infinite branches, the maximal colour that appears infinitely often is even (based on the colouring function $\mathtt{col}'$).*

[4] We slightly abuse notation by identifying a branch with its label.

In Lemma 10, let us explain what we mean by 'Lab_n-projection'. Given an infinite branch

$$s_0 \xrightarrow{\vec{u}_0} (s_0, \mathfrak{f}_0) \rightarrow (\mathfrak{g}_1, s_1) \xrightarrow{\vec{u}_1} (s_1, \mathfrak{f}_1) \rightarrow (\mathfrak{g}_2, s_2) \xrightarrow{\vec{u}_2} (s_2, \mathfrak{f}_2) \rightarrow (\mathfrak{g}_3, s_3) \cdots$$

in a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star}$, its *$Lab_n$-projection* is simply defined as the ω-word in Σ_n^ω below:

$$Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2)\ Lab_n(s_2)\ \cdots$$

Proof. (I) $\rightarrow$ (II) Let $\hat{\mathcal{D}} : \mathfrak{T} \rightarrow Q \times \mathbb{N}^r$ be a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ such that $\hat{\mathcal{D}}(\varepsilon) = (s^\star, \vec{b})$, all the maximal branches are infinite and their Lab_n-projections belong to $\mathrm{L}(\mathbb{A})$. This means that for any label

$$s_0 \xrightarrow{\vec{u}_0} (s_0, \mathfrak{f}_0) \rightarrow (\mathfrak{g}_1, s_1) \xrightarrow{\vec{u}_1} (s_1, \mathfrak{f}_1) \rightarrow (\mathfrak{g}_2, s_2) \xrightarrow{\vec{u}_2} (s_2, \mathfrak{f}_2) \rightarrow \cdots$$

of an infinite branch, we have $Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2)\ Lab_n(s_3)\ \cdots \in \mathrm{L}(\mathbb{A})$.

Let $\hat{\mathcal{D}}' : \mathfrak{T} \rightarrow (Q \times Q') \times \mathbb{N}^r$ be the map defined below. $\hat{\mathcal{D}}'$ will turn out to be a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$ built over the same infinite tree $\mathfrak{T}$. Let $i_1 i_2 i_3 \cdots$ be an infinite branch with the label above such that

$$Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2)\ Lab_n(s_3)\ \cdots \in \mathrm{L}(\mathbb{A}).$$

Since $\mathbb{A}$ is deterministic, there is a unique (accepting) run $q'_0 \xrightarrow{Lab_n(s_0)} q'_1 \xrightarrow{Lab_n(s_1)} q'_2 \cdots$; hence the maximal colour that appears infinitely often is even. For any finite prefix $\mathfrak{w} \sqsubset i_1 i_2 i_3 \cdots$ of length N, we have

$$\hat{\mathcal{D}}'(\mathfrak{w}) \overset{\text{def}}{=} ((q, q'_{\lfloor \frac{N}{2} \rfloor}), \vec{v}) \text{ where } \hat{\mathcal{D}}(\mathfrak{w}) = (q, \vec{v}).$$

Since $\mathbb{A}$ is deterministic, the map $\hat{\mathcal{D}}'$ can be defined uniquely. Indeed, classically, if $\mathbb{A}$ were nondeterministic, we cannot guarantee that two infinite words in $\mathrm{L}(\mathbb{A})$ sharing a common non-empty prefix have the same subrun for that prefix. Since $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$ is also the synchronised product between $\mathcal{A}_{\mathfrak{M},A,s^\star}$ and $\mathbb{A}$, we can check that $\hat{\mathcal{D}}'$ is indeed a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$, such that $\hat{\mathcal{D}}'(\varepsilon) = ((s^\star, q'_0), \vec{b})$ and all the maximal branches are infinite. Consider below the label of any infinite branch:

$$(s_0, q'_0) \xrightarrow{\vec{u}_0} ((s_0, \mathfrak{f}_0), q'_0) \rightarrow ((\mathfrak{g}_1, s_1), q'_1) \xrightarrow{\vec{u}_1} ((s_1, \mathfrak{f}_1), q'_1) \rightarrow ((\mathfrak{g}_2, s_2), q'_2) \xrightarrow{\vec{u}_2} \cdots$$

Since $q'_0 \xrightarrow{Lab_n(s_0)} q'_1 \xrightarrow{Lab_n(s_1)} q'_2 \cdots$ is an accepting run of $\mathbb{A}$, the maximal colour that appears infinitely often on the branch is even.

(II) $\rightarrow$ (I) Let $\hat{\mathcal{D}} : \mathfrak{T} \rightarrow (Q \times Q') \times \mathbb{N}^r$ be a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{A}$ whose root is $((s^\star, q'_0), \vec{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even. Let $\hat{\mathcal{D}}' : \mathfrak{T} \rightarrow Q \times \mathbb{N}^r$ be the map defined below. $\hat{\mathcal{D}}'$ will turn out to be a proof of $\mathcal{A}_{\mathfrak{M},A,s^\star}$, and can be viewed as $\hat{\mathcal{D}}$ where the component in Q' is omitted. For all $\mathfrak{w} \in \mathfrak{T}$, $\hat{\mathcal{D}}'(\mathfrak{w}) \overset{\text{def}}{=} (q, \vec{v})$ where $\hat{\mathcal{D}}(\mathfrak{w}) = ((q, q'), \vec{v})$. We have $\hat{\mathcal{D}}(\varepsilon) = (s^\star, \vec{b})$ and all the maximal branches are infinite. Consider below the label of an infinite branch $i_1 i_2 i_3 \cdots$:

$$s_0 \xrightarrow{\vec{u}_0} (s_0, \mathfrak{f}_0) \rightarrow (\mathfrak{g}_1, s_1) \xrightarrow{\vec{u}_1} (s_1, \mathfrak{f}_1) \rightarrow (\mathfrak{g}_2, s_2) \xrightarrow{\vec{u}_2} (s_2, \mathfrak{f}_2) \rightarrow \cdots$$

Since $\hat{\mathcal{D}}'$ is obtained from $\hat{\mathcal{D}}$ by projection, the label of $i_1 i_2 i_3 \cdots$ in $\hat{\mathcal{D}}$ is of the form

$$(s_0, q'_0) \xrightarrow{\vec{u}_0} ((s_0, \mathfrak{f}_0), q'_0) \rightarrow ((\mathfrak{g}_1, s_1), q'_1) \xrightarrow{\vec{u}_1} ((s_1, \mathfrak{f}_1), q'_1) \rightarrow ((\mathfrak{g}_2, s_2), q'_2) \xrightarrow{\vec{u}_2} \cdots$$

By assumption on $\hat{\mathcal{D}}$, the maximal colour that appears infinitely often is even, and therefore $q'_0 \xrightarrow{Lab_n(s_0)} q'_1 \xrightarrow{Lab_n(s_1)} q'_2 \cdots$ is an accepting run of $\mathbb{A}$ and

$$Lab_n(s_0)\ Lab_n(s_1)\ Lab_n(s_2)\ Lab_n(s_3)\ \cdots \in \mathrm{L}(\mathbb{A}),$$

which concludes the proof. □

6.5. *Decision procedures for RB±ATL* model checking*

In this section, we provide an analysis leading to optimal decision procedures for solving the model checking problem for RB±ATL*, as far as worst-case computational complexity is concerned.

Theorem 6. *The model checking problem for RB±ATL* is decidable.*

Proof. The model checking problem for RB±ATL* is solved by using the algorithm for the parity game problem for alternating VASS (with a simple renaming technique) as a subroutine. The algorithm uses a dynamic programming approach that first computes in which states the subformulae hold before dealing with larger formulae. However, there is a caveat: when dealing with subformulae of the form $\langle\langle A^{\vec{b}}\rangle\rangle\ \Phi$ where Φ is a path formula without any strategy modality, we are entitled to use the algorithm to solve the parity game problem for alternating VASS. However, in order to systematically consider such subformulae $\langle\langle A^{\vec{b}}\rangle\rangle\ \Phi$ when the outermost connective is a strategy modality, we need to perform renamings on-the-fly.

Let ϕ be a formula built over the propositional variables $\{p_1, \dots, p_n\}$ and $\langle\langle A^{\vec{b}}\rangle\rangle\ \Phi$ be one of its subformulae such that no strategy modality occurs in Φ. Without loss of generality, we can assume that there is an injective map $\mathtt{nom} : S \to [1, n]$, such that for every $s \in S$, $Lab(p_{\mathtt{nom}(s)}) = \{s\}$. As a result, each propositional variable $p_{\mathtt{nom}(s)}$ is true in a single state, namely in s. So, even though we assume that ϕ is built over $\{p_1, \dots, p_n\}$, we do not require that all the propositional variables in $\{p_1, \dots, p_n\}$ necessarily occur in ϕ (some of the propositional variables are only used to name states). Given a finite concurrent game structure $\mathfrak{M}$, it is always possible to enrich it so that each state can be named by a dedicated propositional variable (also called a nominal in hybrid logics, see e.g. [7]). This can be done in linear time.

Since Φ is an LTL formula built over $\{p_1, \dots, p_n\}$, there is a Büchi automaton $\mathbb{A}$ over the alphabet Σ_n such that $\mathrm{L}(\mathbb{A})$ is equal to the set of models of Φ (over the set of propositional variables $\{p_1, \dots, p_n\}$), see e.g. [44]. Say that $\mathbb{A}$ has N states and $N \le 2^{|\Phi|}$. Since Büchi automata and deterministic parity automata both recognize the set of ω-regular languages, there is deterministic parity automaton $\mathbb{B}$ with initial location q_0', $O(N!^2)$ states and $2N$ priorities such that $\mathrm{L}(\mathbb{A}) = \mathrm{L}(\mathbb{B})$ [41]. The automaton $\mathbb{B}$ can be effectively computed from $\mathbb{A}$.[5]

Let $X \subseteq S$ be the set of states $s^\star$ such that $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{B}$ has a proof whose root is equal to $((s^\star, q_0'), \vec{b})$, all the maximal branches are infinite and the maximal colour that appears infinitely often is even. We update the formula ϕ by replacing every occurrence of $\langle\langle A^{\vec{b}}\rangle\rangle\ \Phi$ by $\psi = \bigvee_{s' \in X} p_{\mathtt{nom}(s')}$. The set X can be computed thanks to Proposition 4, and this is a correct step thanks to Lemma 10 and Lemma 4. Indeed, for all $s' \in S$, we have $\mathfrak{M}, s' \models_{\mathfrak{s}} \langle\langle A^{\vec{b}}\rangle\rangle\ \Phi$ iff $\mathfrak{M}, s' \models_{\mathfrak{s}} \bigvee_{s' \in X} p_{\mathtt{nom}(s')}$, and, therefore, for all $s' \in S$, we have $\mathfrak{M}, s' \models_{\mathfrak{s}} \phi$ iff $\mathfrak{M}, s' \models_{\mathfrak{s}} \phi[\psi / \langle\langle A^{\vec{b}}\rangle\rangle\ \Phi]$, where $\phi[\psi / \langle\langle A^{\vec{b}}\rangle\rangle\ \Phi]$ is obtained from ϕ by substituting every occurrence of $\langle\langle A^{\vec{b}}\rangle\rangle\ \Phi$ by ψ. We update ϕ until there are no more strategy modalities, and therefore eventually ϕ is a Boolean combination of propositional variables, which is then easy to evaluate on a given state. It is worth noting that the total number of calls to the parity game problem for AVASS is linear in the size of the formula, each instance of the problem has a doubly-exponential number of locations, and the colouring map has an exponential number of priorities in the size of the input formula. □

The proof of Theorem 6 uses a synchronised product between an alternating VASS and a deterministic parity automaton recognising ω-words. This is reminiscent of the proof of a 2EXPTIME upper bound for ATL* model checking problem [6, Theorem 5.6]. However, the Rabin tree automata in the proof of [6, Theorem 5.6] are replaced by deterministic parity automata for encoding the LTL formulae, and by alternating VASS (with counters) as outcome of the synchronisation.

Moreover, using the very recent developments in [16], a 2EXPTIME upper bound can be obtained too.

Theorem 7. *The model checking problem for RB±ATL* is in 2EXPTIME.*

Proof. The complexity upper bound is obtained by using the algorithm described in the proof of Theorem 6, by analysing the size of the instances solved for the parity game problem for AVASS and then to invoke [16, Corollary 5.7], briefly recalled below.

Let us make a few simple observations from the proof of Theorem 6.

1. The number of calls to the subroutine solving the parity game problem for AVASS is linear in the size of the input formula.
2. Each AVASS $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{B}$ built in the proof verifies the following quantitative properties.
   (a) The size of $\mathcal{A}_{\mathfrak{M},A,s^\star}$ is polynomial in the size of the input concurrent game structure $\mathfrak{M}$.
   (b) As far as the deterministic parity automaton $\mathbb{B}$ is concerned, the number of states is doubly-exponential in the size of the input formula and the number of priorities is exponential in the size of the input formula.
   (c) The number of states in $\mathcal{A}_{\mathfrak{M},A,s^\star} \otimes \mathbb{B}$ is doubly-exponential in the size of the inputs (including $\mathfrak{M}$ and ϕ) and each vector has values at most exponential in the size of $\mathfrak{M}$.

We have seen that instances of the parity game problem for AVASS can be reduced to instances of the parity game problem for single-sided VASS, which itself can be turned into instances of the energy parity game, see e.g. [1, Lemma 4] and these two reductions can be performed without changing significantly the size of the instances as well as the maximal values in vectors.

[5] A similar construction to that for $\mathbb{B}$ was used recently in [15] for model checking pushdown multi-agent systems.

According to [16, Corollary 5.7], an instance of the energy game problem with initial credit $\vec{c}$, with game graph (V, E, r) ($V = V_1 \uplus V_2$ and E is a finite set of edges in $V \times \mathbb{Z}^r \times V$) and priority function $\pi : V \to \mathbb{N}$ with $p = \mathrm{card}(\{\pi(v) \mid v \in V\})$, is solvable in time

$$(\mathrm{card}(V) \times \| E \|)^{2^{O(r \times log(r+p))}} + O(r \times log \| \vec{c} \|),$$

where

- $\| \vec{w} \| = \max\{| \vec{w}(i) | : i \in [1, r]\}$ for all $\vec{w} \in \mathbb{Z}^r$,
- $\| E \| = \max\{\| \vec{w} \| : v \xrightarrow{\vec{w}} v' \in E\}$.

So, all the instances of the energy parity games that we consider in the decidability satisfy the following properties:

- $\mathrm{card}(V)$ is doubly-exponential in the size of the inputs,
- $\| E \|$ is exponential in the size of the input concurrent game structure,
- r is unchanged,
- p is exponential in the size of the input formula,
- $\| \vec{c} \|$ is exponential in the size of the input formula.

Consequently, each instance of the energy parity game problem can be solved in doubly-exponential time, and therefore each instance of the parity game problem for AVASS can be solved in doubly-exponential time, which leads to a total doubly-exponential time since the number of calls is linear in the size of the input formula. □

Note that although RB±ATL and RB±ATL* have identical worst-case computational complexity (namely 2EXPTIME-completeness) in order to solve the model-checking problem for RB±ATL, we only need to call subroutines for the state reachability and non-termination problems for AVASS whereas RB±ATL* requires calls to subroutines to the more general parity game problem for AVASS. The restriction to a single agent also leads to an EXPSPACE upper bound and this is optimal.

Theorem 8. *The model checking problem for RB±ATL* restricted to a single agent is* EXPSPACE*-complete.*

The proof is similar to the proof of Theorem 4. Again, the upper bound is obtained by solving instances of the model checking problem for LTL on VASS (when the coalition is a singleton set) or instances of the model checking problem for CTL* (when the coalition is the empty set), which are known to be EXPSPACE-complete (see e.g. [28]) and PSPACE-complete (see e.g. [42]) respectively.

Note also that resource-bounded concurrent game structures can be seen as generalisations of VASS and, the logic RB±ATL* is clearly a generalisation of CTL*, by using the correspondences $\langle\langle Agt^{\vec{\omega}} \rangle\rangle\ \Phi \approx \mathsf{E}\ \Phi$, and $\langle\langle \emptyset^{\vec{\omega}} \rangle\rangle\ \Phi \approx \mathsf{A}\ \Phi$. It may seem surprising that the model checking problem for RB±ATL* is decidable, given that the model checking problem for CTL* on VASS is known to be undecidable, see e.g. [25]. However this can be explained by the different satisfaction relations in the two problems. In the case of RB±ATL*, formulae are evaluated on states of a concurrent game structure, not on configurations made of states and counter values, and this makes all the difference.

It is also remarkable that the proof of Theorem 6 does *not* use the fact that the idle action is always among the action(s) returned by the action manager. In contrast, the proofs in Section 4 use the idle action in order to extend finite computations to infinite ones, by choosing the idle action for all the agents after the finite part of the computation that, e.g., witnesses the satisfaction of a next or an until formula (see, e.g., Lemma 8). This difference can be explained by the fact that to solve the model-checking for RB±ATL*, we use a subroutine to a more general decision problem (the parity game problem for alternating VASS) and therefore one can be a bit more liberal on the conditions satisfied by the concurrent game structures. As a consequence, we get the following decidability result (and the results in Section 6.6 below also hold for RB±ATL* without idle actions).

Corollary 3. *The model checking problem for the variant of RB±ATL* in which no idle action is assumed in the resource-bounded concurrent game structures (and the action manager always returns a non-empty set of actions) is in* 2EXPTIME.

The proof is the same as for Theorem 6 and Theorem 8. The restriction to a single agent can be also shown EXPSPACE-complete, by using analogous arguments from the proof of Theorem 8 (EXPSPACE-hardness proof is even simpler).

By combining our results from previous sections and those from [1], we have shown that the model checking problem for RB±ATL* is decidable. However, by exploiting techniques for the effective computation of the Pareto frontiers from [1], we can go further, and actually synthesise values for parameters. This is the subject of the next section.

6.6. Symbolic representations for sets of resource values

Let $\mathfrak{M} = (Agt, S, Act, r, \mathtt{act}, \mathtt{cost}, \delta, Lab)$ be a resource-bounded concurrent game structure and ϕ be a ParRB±ATL* formula such that its resource variables are among $\mathtt{x}_1, \dots, \mathtt{x}_m$ and its propositional variables are among $p_1, \dots, p_n$.

We write $\varphi_1 = \langle\langle A_1^{\mathtt{t}_1}\rangle\rangle\phi_1, \dots, \varphi_\alpha = \langle\langle A_\alpha^{\mathtt{t}_\alpha}\rangle\rangle\phi_\alpha$ to denote the subformulae of ϕ whose outermost connective is a strategy modality. The subformulae are arranged in order of increasing size. So, ϕ_1 does not contain a strategy modality, and it can be viewed as an LTL formula built over $\{p_1, \dots, p_n\}$. Each expression $\mathtt{t}_i$ is a tuple of r variables, say $\mathtt{t}_i = (\mathtt{t}_i^1, \dots, \mathtt{t}_i^r)$. By definition, a variable can occur more than once in $\mathtt{t}_i$, and two distinct tuples $\mathtt{t}_i$ and $\mathtt{t}_j$ can share variables. This provides great flexibility in the logical formalism. Below, for each state $s \in S$ and $i \in [1, \alpha]$, we build a *constrained formula* $\hat{\varphi}_i^s$ over $\mathtt{x}_1$, …, $\mathtt{x}_m$ following the grammar:

$$\psi ::= \mathtt{x} \geq c \mid \mathtt{x} = \omega \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi,$$

where $\mathtt{x} \in \mathtt{VAR}$ and $c \in \mathbb{N}$. Such formulae are interpreted over valuations $\mathfrak{v} : \mathtt{VAR} \to \mathbb{N} \cup \{\omega\}$ with semantics based on the satisfaction relation $\mathfrak{v} \models \psi$, and have the following key property:

$$\text{for all } \mathfrak{v}, \text{ we have } \mathfrak{v} \models \hat{\varphi}_i^s \text{ iff } \mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i).$$

Note that ϕ_i may contain variables in $\mathtt{x}_1, \dots, \mathtt{x}_m$. So, $\hat{\varphi}_i^s$ characterizes exactly the set of parameter values so that $\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i$ is satisfied on the state s.

Before explaining how to construct the formulae $\hat{\varphi}_i^s$, we first explain how to construct from such formulae $\hat{\varphi}_i^s$ a constrained formula ψ_s such that for all $\mathfrak{v}$, we have $\mathfrak{v} \models \psi_s$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\phi)$. Let φ_{i_1}, …, φ_{i_z} be the maximal subformulae of ϕ such that their outermost connective is a strategy modality (with $\{i_1, \dots, i_z\} \subseteq [1, \alpha]$). Given a propositional valuation $\mathfrak{h} : \{\varphi_{i_1}, \dots, \varphi_{i_z}\} \to \{\bot, \top\}$, we define $\mathfrak{M}, s \models \mathfrak{h}(\phi)$ iff $\mathfrak{h}(\phi)$ obtained from ϕ by replacing simultaneously each φ_{i_j} by $\mathfrak{h}(\varphi_{i_j})$ is true in s ($\mathfrak{h}(\phi)$ may also contain propositional variables). The constrained formula ψ_s is defined as follows:

$$\bigvee_{\mathfrak{h} \text{ s.t. } \mathfrak{M},s \models \mathfrak{h}(\phi)} \Big(\bigwedge_{\ell \text{ s.t. } \mathfrak{h}(\varphi_{i_\ell}) = \top} \hat{\varphi}_{i_\ell}^s \Big) \wedge \Big(\bigwedge_{\ell \text{ s.t. } \mathfrak{h}(\varphi_{i_\ell}) = \bot} \neg\hat{\varphi}_{i_\ell}^s \Big).$$

The generalised disjunction considers all possible valuations $\mathfrak{h}$ that make $\mathfrak{h}(\phi)$ true in s, and the subsequent conjunction ensures that if $\mathfrak{h}(\varphi_\ell) = \top$ then $\hat{\varphi}_\ell^s$ is satisfied, otherwise $\neg\hat{\varphi}_\ell^s$ has to be satisfied.

Now, we explain how to build $\hat{\varphi}_i^s$. Let φ_{j_1}, …, φ_{j_β} be the maximal subformulae of ϕ_i where the outermost connective is a strategy modality. By assumption, the formulae are arranged in order of increasing size, so we have $\{j_1, \dots, j_\beta\} \subseteq [1, i-1] \subset [1, \alpha]$, and, possibly, there may be no such subformulae in the case where ϕ_i has no strategy modality (for instance this happens when $i = 1$).

Given $S_1, \dots, S_\beta \subseteq S$ and $I \subseteq [1, r]$, let $\psi_{I,S_1,\dots,S_\beta}^s$ be the formula encoding the Pareto frontier of $\mathcal{A}_{\mathfrak{M},A_i,s} \otimes \mathbb{A}$ with initial state (s, q_0'), the ω-components are exactly in I. Furthermore, $\mathbb{A}$ is a deterministic parity automaton such that $\mathrm{L}(\mathbb{A})$ is the ω-regular language over the alphabet $\mathcal{P}(\{p_1, \dots, p_n\})$ defined by the LTL formula ϕ_i' that is obtained from ϕ_i by replacing every occurrence of φ_j by

$$\Big(\bigvee_{s' \in S_j} p_{\mathrm{nom}(s')}\Big) \wedge \Big(\bigwedge_{s' \in S \setminus S_j} \neg p_{\mathrm{nom}(s')}\Big).$$

The formula $\hat{\varphi}_i^s$ is then defined as the following disjunction:

$$\bigvee_{S_1,\dots,S_\beta \subseteq S, I \subseteq [1,r]} \psi_{I,S_1,\dots,S_\beta}^s \wedge \Big(\bigwedge_{\gamma \in [1,\beta]} \Big(\bigwedge_{s' \in S_\gamma} \hat{\varphi}_{j_\gamma}^{s'} \Big) \wedge \Big(\bigwedge_{s' \in S \setminus S_\gamma} \neg\hat{\varphi}_{j_\gamma}^{s'} \Big)\Big).$$

The proof of Lemma 11 below provides the formal justification for such a construction. However, intuitively, in the generalised disjunction each S_γ allows us to guess where φ_{j_γ} is true, and therefore $s' \in S_\gamma$ should be equivalent to the satisfaction of $\hat{\varphi}_{j_\gamma}^{s'}$. Since $\{j_1, \dots, j_\beta\} \subseteq [1, i-1]$, each formula $\hat{\varphi}_{j_\gamma}^{s'}$ is already defined, and therefore the formula $\hat{\varphi}_i^s$ can be safely built. (Below, a valuation $\mathfrak{v}$ is also called a *concretisation*.)

Lemma 11. *For all $i \in [1, \alpha]$, for all $s \in S$, for all concretisations $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}_i^s$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i)$.*

Proof. The proof is by induction on i.
Base case: there is no strategy modality in ϕ_i (this includes the case where $i = 1$).

Recall that $\varphi_i = \langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i$ where $A_i \subseteq Agt$ and $\mathtt{t}_i = (\mathtt{t}_i^1, \dots, \mathtt{t}_i^r)$. Since there is no strategy modality in ϕ_i, the formula ϕ_i is simply an LTL formula built over propositional variables in $\{p_1, \dots, p_n\}$. Let $\mathbb{A}$ be a deterministic parity automaton such that the models of ϕ_i on Σ_n are precisely L($\mathbb{A}$). It is known that $\mathbb{A}$ can be effectively computed from ϕ_i such that the number of states in $\mathbb{A}$ is doubly-exponential in the size of ϕ_i, and the number of priorities is (only) exponential in the size of ϕ_i.

Given $I \subseteq [1, r]$, we can construct a constrained formula ψ^s_I characterising the Pareto frontier of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ with initial state (s, q'_0) and the values at the positions in I are equal to ω at the root (see [1, Theorem 4]). Consequently ψ^s_I is equivalent to $\psi^s_I \wedge (\bigwedge_{j\in I} \mathtt{t}^j_i = \omega) \wedge (\bigwedge_{j\notin I} \mathtt{t}^j_i \neq \omega)$. This means that for all $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I) for all $j \in [1, r]$, $\vec{b}(j) = \omega$ iff $j \in I$, we have $\vec{b} \models \psi^s_I$ (meaning $\mathfrak{v} \models \psi^s_I$ with $\mathfrak{v}(\mathtt{t}^j_i) \overset{\text{def}}{=} \vec{b}(j)$ for all $j \in [1, r]$) iff there is a proof of $\mathcal{A}_{\mathfrak{M},A,s} \otimes \mathbb{A}$ whose root is $((s, q'_0), \vec{b})$ and the proof satisfies the parity condition.

This is the place where we use all the previous results.

- By Lemma 10, for all $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I), we have $\vec{b} \models \psi^s_I$ iff $\mathcal{A}_{\mathfrak{M},A_i,s}$ has a proof whose root is $(s, \vec{b})$, all the maximal branches are infinite and the Lab_n-projection of each infinite branch belongs to L($\mathbb{A}$).
- By Lemma 4, for all $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I), we have $\vec{b} \models \psi^s_I$ iff there is a $\vec{b}$-strategy F_{A_i} w.r.t. s in $\mathfrak{M}$ such that the set of computations $\mathtt{Comp}(s, F_{A_i})$ is included in L($\mathbb{A}$).
- So, for all $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ such that (C_I), we have $\vec{b} \models \psi^s_I$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \langle\langle A^{\vec{b}}_i \rangle\rangle \phi_i$.

Consequently, for all concretisations $\mathfrak{v}$ such that for all $j \in [1, r]$ $\mathfrak{v}(\mathtt{t}^j_i) = \omega$ iff $j \in I$, we have $\mathfrak{v} \models \psi^s_I$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A^{\mathtt{t}_i}_i \rangle\rangle \phi_i)$. The formula $\hat{\varphi}^s_i$ is defined as a generalised disjunction parameterised by all the possible values for I, i.e. $\hat{\varphi}^s_i \overset{\text{def}}{=} \bigvee_{I \subseteq [1,r]} \psi^s_I$, and it is easy to check that for all $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}^s_i$ is equivalent to $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A^{\mathtt{t}_i}_i \rangle\rangle \phi_i)$.

Induction step. $\varphi_i = \langle\langle A^{\mathtt{t}_i}_i \rangle\rangle \phi_i$ and $\varphi_{j_1}, \ldots, \varphi_{j_\beta}$ are the maximal subformulae of ϕ_i ($\beta \geq 1$) such that its outermost connective is a strategy modality and for all $\gamma \in [1, \beta]$, for all $s' \in S$, for all $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}^{s'}_{j_\gamma}$ iff $\mathfrak{M}, s' \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A^{\mathtt{t}_{j_\gamma}}_{j_\gamma} \rangle\rangle \phi_{j_\gamma})$.

The proof for the induction step is quite similar to the proof for the base case, except that we need to show that the renaming mechanism we use is correct. First, let us state a few basic properties. Given a state formula ϕ in RB±ATL*, we write $\mathfrak{M} \models \phi \overset{\text{def}}{\Leftrightarrow}$ for all $s' \in S$, we have $\mathfrak{M}, s' \models_{\mathfrak{s}} \phi$, i.e. ϕ is valid in $\mathfrak{M}$.

(P1) Let ϕ, ψ, ψ' be state formulae in RB±ATL* such that ψ occurs in ϕ and $\mathfrak{M} \models \psi \Leftrightarrow \psi'$. Then $\mathfrak{M} \models \phi \Leftrightarrow \phi[\psi'/\psi]$, where $\phi[\psi'/\psi]$ is defined from ϕ by replacing every occurrence of ψ by ψ'.

(P2) Let $\gamma \in [1, \beta]$ and $S' \subseteq S$. For all $\mathfrak{v}$, the statements below are equivalent.

(P2.1) $\mathfrak{M} \models [(\bigvee_{s'\in S'} p_{\mathrm{nom}(s')}) \wedge (\bigwedge_{s'\notin S'} \neg p_{\mathrm{nom}(s')})] \Leftrightarrow \mathfrak{v}(\varphi_{j_\gamma})$.

(P2.2) For all $s' \in S$, $\mathfrak{v} \models \hat{\varphi}^{s'}_{j_\gamma}$ iff $s' \in S'$.

The proof of (P1) is quite standard but it is worth noting that the formulae ϕ, ψ and ψ' need to be *state formulae*. The proof of (P2) uses the induction hypothesis in a straightforward way.

Let $S_1, \ldots, S_\beta \subseteq S$ and $I \subseteq [1, r]$. We write ϕ'_i to denote the formula obtained from ϕ_i replacing every occurrence of the formula φ_{j_γ} by

$$\varphi^\star_{j_\gamma} \overset{\text{def}}{=} (\bigvee_{s'\in S_\gamma} p_{\mathrm{nom}(s')}) \wedge (\bigwedge_{s'\in S\setminus S_\gamma} \neg p_{\mathrm{nom}(s')}).$$

So, even though this is not explicit in the notation, the formula ϕ'_i obviously depends on $S_1, \ldots, S_\beta$. Like the base case, the formula ϕ'_i is an LTL formula built over $\{p_1, \ldots, p_n\}$, and one can compute a deterministic parity automaton $\mathbb{A}$ such that the models of ϕ'_i on Σ_n are precisely L($\mathbb{A}$). Again, we can construct a constrained formula $\psi^s_{I,S_1,\ldots,S_\beta}$ characterising the Pareto frontier of $\mathcal{A}_{\mathfrak{M},A_i,s} \otimes \mathbb{A}$, with initial state (s, q'_0) and the values at the positions in I are equal to ω at the root, see [1, Theorem 4].

Reasoning in the same way as in the base case (basically replace ϕ_i by ϕ'_i), we can show that, for all concretisations $\mathfrak{v}$ such that for all $j \in [1, r]$ $\mathfrak{v}(\mathtt{t}^j_i) = \omega$ iff $j \in I$, we have $\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta}$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A^{\mathtt{t}_i}_i \rangle\rangle \phi'_i)$ (†). However, the formula that is important to us is $\langle\langle A^{\mathtt{t}_i}_i \rangle\rangle \phi_i$ (rather than $\langle\langle A^{\mathtt{t}_i}_i \rangle\rangle \phi'_i$), and this is the place where the induction hypothesis is again invoked.

First, let us introduce an auxiliary notion. A concretisation $\mathfrak{v}$ is said to be *compatible with* $I, S_1, \ldots, S_\beta$ iff the conditions below hold:

1. $I = \{\gamma \in [1, r] \mid \mathfrak{v}(\mathtt{t}^\gamma_i) = \omega\}$.
2. For all $\gamma \in [1, \beta]$, $S_\gamma = \{s' \in S \mid \mathfrak{M}, s' \models_{\mathfrak{s}} \mathfrak{v}(\varphi_{j_\gamma})\}$.

Assuming that $\mathfrak{v}$ is compatible with $I, S_1, \ldots, S_\beta$, we have that:

- By induction hypothesis, for all $\gamma \in [1, \beta]$, $S_\gamma = \{s' \in S \mid \mathfrak{v} \models \hat{\varphi}^{s'}_{j_\gamma}\}$.
- By (P2), for all $\gamma \in [1, \beta]$, $\mathfrak{M} \models [(\bigvee_{s'\in S_\gamma} p_{\mathrm{nom}(s')}) \wedge (\bigwedge_{s'\notin S_\gamma} \neg p_{\mathrm{nom}(s')})] \Leftrightarrow \mathfrak{v}(\varphi_{j_\gamma})$.

- By (P1) and the newly established property (†) – see above, –

$$\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta} \text{ iff } \mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi'_i)[\varphi_{j_1}/\varphi^\star_{j_1}, \ldots, \varphi_{j_\beta}/\varphi^\star_{j_\beta}].$$

- Since $\mathfrak{v}(\phi'_i)[\varphi_{j_1}/\varphi^\star_{j_1}, \ldots, \varphi_{j_\beta}/\varphi^\star_{j_\beta}] = \mathfrak{v}(\phi_i)$, $\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta}$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i)$.
- By using the compatibility of $\mathfrak{v}$, we get that $\mathfrak{v} \models \psi^s_{I,S_1,\ldots,S_\beta} \wedge (\bigwedge_{\gamma\in[1,\beta]}(\bigwedge_{s'\in S_\gamma} \hat{\varphi}^{s'}_{j_\gamma}) \wedge (\bigwedge_{s'\notin S_\gamma} \neg\hat{\varphi}^{s'}_{j_\gamma}))$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i)$.

The formula $\hat{\varphi}^s_i$ is defined as a generalised disjunction parameterised by all the possible values for I, S_1, …, S_β, and it is easy to check that for all $\mathfrak{v}$, we have $\mathfrak{v} \models \hat{\varphi}^s_i$ equivalent to $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_i^{\mathtt{t}_i}\rangle\rangle\phi_i)$. □

Theorem 9. *The parameterised model checking problem for ParRB±ATL* can be solved.*

It is worth noting that the size of the symbolic representations is very large in the worst case since the construction of the representation of the Pareto frontier for parity games on single-sided VASS in [1] is based on some algorithm that is similar to Karp–Miller algorithm for Petri nets.

Proof. Let ϕ be a state formula in ParRB±ATL*$([1,k], r)$, $\mathfrak{M}$ be a resource-bounded concurrent game structure and $s \in S$. We write $\varphi_1 = \langle\langle A_1^{\mathtt{t}_1}\rangle\rangle\phi_1$, …, $\varphi_z = \langle\langle A_z^{\mathtt{t}_z}\rangle\rangle\phi_z$ to denote the maximal subformulae of ϕ such that the outermost connective is a strategy modality. By Lemma 11, for all $j \in [1,z]$, for all valuations $\mathfrak{v} : \mathrm{VAR} \to \mathbb{N} \cup \{\omega\}$, we have $\mathfrak{v} \models \hat{\varphi}^s_j$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\langle\langle A_j^{\mathtt{t}_j}\rangle\rangle\phi_j)$, where $\hat{\varphi}^s_j$ is a constrained formula that can be built from ϕ, $\mathfrak{M}$ and s. Let ψ_s be the formula defined by:

$$\bigvee_{\mathfrak{h}:\{\varphi_1,\ldots,\varphi_z\}\to\{\bot,\top\} \text{ s.t. } \mathfrak{M},s\models\mathfrak{h}(\phi)} \Big(\bigwedge_{j \text{ s.t. } \mathfrak{h}(\varphi_j)=\top} \hat{\varphi}^s_j\Big) \wedge \Big(\bigwedge_{j \text{ s.t. } \mathfrak{h}(\varphi_j)=\bot} \neg\hat{\varphi}^s_j\Big).$$

Each expression $\hat{\varphi}^s_j$ and $\neg\hat{\varphi}^s_j$ is a constrained formula, and $\mathfrak{M}, s \models \mathfrak{h}(\phi)$ can be decided for any valuation $\mathfrak{h}$. Consequently, ψ_s defined above is a constrained formula that can be effectively computed.

Let $\mathfrak{v}$ be a concretisation such that $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\phi)$. There exists $\mathfrak{h}_0 : \{\varphi_1, \ldots, \varphi_z\} \to \{\bot, \top\}$ such that $\mathfrak{M}, s \models \mathfrak{h}_0(\phi)$ and for all $j \in [1,z]$, we have $\mathfrak{h}_0(\varphi_j) = \top$ iff $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\varphi_j)$. By Lemma 11, for all $j \in [1,z]$, we have $\mathfrak{h}_0(\varphi_j) = \top$ iff $\mathfrak{v} \models \hat{\varphi}^s_j$. So,

$$\mathfrak{v} \models \Big(\bigwedge_{j \text{ s.t. } \mathfrak{h}_0(\varphi_j)=\top} \hat{\varphi}^s_j\Big) \wedge \Big(\bigwedge_{j \text{ s.t. } \mathfrak{h}_0(\varphi_j)=\bot} \neg\hat{\varphi}^s_j\Big),$$

which entails $\mathfrak{v} \models \psi_s$.

Conversely, assuming that $\mathfrak{v} \models \psi_s$ for some concretisation $\mathfrak{v}$, there is a map $\mathfrak{h}$ such that $\mathfrak{v} \models (\bigwedge_{j \text{ s.t. } \mathfrak{h}(\varphi_j)=\top} \hat{\varphi}^s_j) \wedge (\bigwedge_{j \text{ s.t. } \mathfrak{h}(\varphi_j)=\bot} \neg\hat{\varphi}^s_j)$ and $\mathfrak{M}, s \models \mathfrak{h}(\phi)$. Again by Lemma 11, and by Boolean reasoning we obtain $\mathfrak{M}, s \models \mathfrak{v}(\phi)$. Consequently, the formula ψ_s can be computed and it is a symbolic representation for all the maps $\mathfrak{v}$ such that $\mathfrak{M}, s \models_{\mathfrak{s}} \mathfrak{v}(\phi)$. □

7. Concluding remarks

We have related model checking problems for resource-bounded logics and decision problems for alternating VASS, such as state reachability, non-termination, and, more generally, parity game problems. While the existence of such relationships is perhaps not surprising, we have been able to obtain several new complexity and decidability results, as recalled below.

1. The model checking problem for the logic RB±ATL introduced in [4,5] is 2EXPTIME-complete. No complexity upper bound was previously known. The complexity upper bound is obtained by using the subroutines to solve respectively the state reachability and non-termination problems for AVASS.
2. We have introduced a new logic RB±ATL* that extends RB±ATL (as ATL* extends ATL), and we have shown that its model checking problem is decidable by using the subroutine for the parity game problem for AVASS. Recent developments in [16] allowed us to refine this result to an 2EXPTIME upper bound too. For the parameterised version ParRB±ATL*, given $\mathfrak{M}$, s and ϕ in ParRB±ATL*, we have explained how we can synthetise a formula ψ such that $\mathfrak{M}, s \models \mathfrak{v}(\phi)$ iff $\mathfrak{v} \models \psi$ for all interpretations $\mathfrak{v}$ for the resource parameters. Moreover, ψ is a Boolean combination of atomic constraints of the form $\mathtt{x} \geq k$ and $\mathtt{x} = \omega$. A summary of the main complexity results for the model-checking problems can be found below.

	No restriction	card$(Agt) = 1$	fixed $r = 2$	fixed $r \geq 4$
RB±ATL	2EXPTIME-c.	EXPSPACE-c.	PSPACE-h., in EXPTIME	EXPTIME-c.
RB±ATL*	2EXPTIME-c.	EXPSPACE-c.	PSPACE-h.	EXPTIME-h.

3. The model checking problem for RBTL* introduced in [12] is EXPSPACE-complete and when the number of resources r is fixed and greater to 2, the problem is PSPACE-complete (see details in Section 5.2). The decidability of the model checking problem for RBTL* and the complexity upper bound for RBTL were not previously known.

We have been also able to provide complexity results for fragments and variants of these resource-bounded logics, and we believe that the simple framework we have proposed may be used to obtain further results for new resource-bounded logics. However this is future work.

Acknowledgements

We would like to thank the anonymous referees for their remarks and suggestions that helped us a lot to improve the quality of the document.

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

References

[1] P.A. Abdulla, R. Mayr, A. Sangnier, J. Sproston, Solving parity games on integer vectors, in: CONCUR'13, in: Lecture Notes in Computer Science, vol. 8052, Springer, 2013, pp. 106–120.
[2] N. Alechina, N. Bulling, S. Demri, B. Logan, On the complexity of resource-bounded logics, in: RP'16, in: Lecture Notes in Computer Science, vol. 9899, Springer, 2016, pp. 36–50.
[3] N. Alechina, N. Bulling, B. Logan, H.N. Nguyen, On the boundary of (un)decidability: decidable model-checking for a fragment of resource agent logic, in: IJCAI'15, AAAI Press, 2015, pp. 1494–1501.
[4] N. Alechina, B. Logan, H.N. Nguyen, F. Raimondi, Decidable model-checking for a resource logic with production of resources, in: ECAI'14, 2014, pp. 9–14.
[5] N. Alechina, B. Logan, H.N. Nguyen, F. Raimondi, Technical report: model-checking for resource-bounded ATL with production and consumption of resources, CoRR, arXiv:1504.06766, 2015.
[6] R. Alur, Th. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (5) (2002) 672–713.
[7] C. Areces, P. Blackburn, M. Marx, Hybrid logics: characterization, interpolation and complexity, J. Symbolic Logic 66 (3) (2001) 977–1010.
[8] M. Blockelet, S. Schmitz, Model-checking coverability graphs of vector addition systems, in: MFCS'11, in: Lecture Notes in Computer Science, vol. 6907, Springer, 2011, pp. 108–119.
[9] M. Blondin, A. Finkel, S. Göller, C. Haase, P. McKenzie, Reachability in two-dimensional vector addition systems with states is PSPACE-complete, in: LICS'15, ACM Press, 2015, pp. 32–43.
[10] A. Bouajjani, M. Bozga, P. Habermehl, R. Iosif, P. Moro, T. Vojnar, Programs with lists are counter automata, in: CAV'06, in: Lecture Notes in Computer Science, vol. 4144, Springer, 2006, pp. 517–531.
[11] T. Brázdil, P. Jančar, A. Kucera, Reachability games on extended vector addition systems with states, in: ICALP'10, in: Lecture Notes in Computer Science, vol. 6199, Springer, 2010, pp. 478–489.
[12] N. Bulling, B. Farwer, Expressing properties of resource-bounded systems: the logics RBTL* and RBTL, in: CLIMA X, in: Lecture Notes in Computer Science, vol. 6214, Springer, 2009, pp. 22–45.
[13] N. Bulling, B. Farwer, On the (un-)decidability of model-checking resource-bounded agents, in: ECAI'10, 2010, pp. 567–572.
[14] N. Bulling, H.N. Nguyen, Model checking resource bounded systems with shared resources via alternating Büchi pushdown systems, in: PRIMA'15, in: Lecture Notes in Computer Science, vol. 9387, 2015, pp. 640–649.
[15] T. Chen, F. Song, Z. Wu, Global model checking on pushdown multi-agent systems, in: AAAI'16, AAAI Press, 2016, pp. 2459–2465.
[16] Th. Colcombet, M. Jurdziński, R. Lazić, S. Schmitz, Perfect half space games, in: LICS'17, IEEE Press, 2017, pp. 1–11.
[17] J.B. Courtois, S. Schmitz, Alternating vector addition systems with states, in: MFCS'14, in: Lecture Notes in Computer Science, vol. 8634, Springer, 2014, pp. 220–231.
[18] Ph. Darondeau, S. Demri, R. Meyer, Ch. Morvan, Petri net reachability graphs: decidability status of FO properties, Log. Methods Comput. Sci. 8 (4) (2012).
[19] S. Demri, On selective unboundedness of VASS, J. Comput. System Sci. 79 (5) (2013) 689–713.
[20] S. Demri, V. Goranko, M. Lange, Temporal Logics in Computer Science, Cambridge University Press, 2016.
[21] S. Demri, M. Jurdziński, O. Lachish, R. Lazić, The covering and boundedness problems for branching vector addition systems, J. Comput. System Sci. 79 (1) (2013) 23–38.
[22] L.E. Dickson, Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors, Amer. J. Math. (1913) 413–422.
[23] A. Emerson, Temporal and modal logic, in: Handbook of Theoretical Computer Science, Elsevier, 1990, pp. 996–1072.
[24] J. Esparza, On the decidability of model checking for several μ-calculi and Petri nets, in: ICALP'94, in: Lecture Notes in Computer Science, vol. 787, Springer, 1994, pp. 115–129.
[25] J. Esparza, Decidability and complexity of Petri net problems — an introduction, in: Advances in Petri Nets 1998, in: Lecture Notes in Computer Science, vol. 1491, Springer, 1998, pp. 374–428.
[26] S. Göller, M. Lohrey, Branching-time model checking of one-counter processes and timed automata, SIAM J. Comput. 42 (3) (2013) 884–923.
[27] C. Haase, On the Complexity of Model Checking Counter Automata, PhD thesis, University of Oxford, 2012.
[28] P. Habermehl, On the complexity of the linear-time mu-calculus for Petri nets, in: Lecture Notes in Computer Science, vol. 1248, Springer, 1997, pp. 102–116.
[29] R.R. Howell, L.E. Rosier, Problems concerning fairness and temporal logic for conflict-free Petri nets, Theoret. Comput. Sci. 64 (1989) 305–329.
[30] P. Jančar, Decidability of a temporal logic problem for Petri nets, Theoret. Comput. Sci. 74 (1) (1990) 71–93.
[31] P. Jančar, On reachability-related games on vector addition systems with states, in: RP'15, in: Lecture Notes in Computer Science, vol. 9328, Springer, 2015, pp. 50–62.

[32] L. Juhl, K.G. Larsen, J.-F. Raskin, Optimal bounds for multiweighted and parametrised energy games, in: Theories of Programming and Formal Methods – Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday, in: Lecture Notes in Computer Science, vol. 8051, Springer, 2013, pp. 244–255.

[33] M. Jurdziński, R. Lazić, S. Schmitz, Fixed-dimensional energy games are in pseudo-polynomial time, in: ICALP'15, in: Lecture Notes in Computer Science, vol. 9135, Springer, 2015, pp. 260–272.

[34] R.M. Karp, R.E. Miller, Parallel program schemata, J. Comput. System Sci. 3 (2) (1969) 147–195.

[35] F. Laroussinie, N. Markey, G. Oreiby, On the expressiveness and complexity of ATL, Log. Methods Comput. Sci. 4 (2) (2008).

[36] J. Leroux, S. Schmitz, Demystifying reachability in vector addition systems, in: LICS'15, IEEE, 2015, pp. 56–67.

[37] R.J. Lipton, The Reachability Problem Requires Exponential Space, Technical Report 62, Department of Computer Science, Yale University, 1976.

[38] D. Della Monica, M. Napoli, M. Parente, On a logic for coalitional games with priced-resource agents, Electron. Notes Theor. Comput. Sci. 278 (2011) 215–228.

[39] C. Rackoff, The covering and boundedness problems for vector addition systems, Theoret. Comput. Sci. 6 (2) (1978) 223–231.

[40] J.F. Raskin, M. Samuelides, L. Van Begin, Games for counting abstractions, Electron. Notes Theor. Comput. Sci. 128 (6) (2005) 69–85.

[41] S. Schewe, Tighter bounds for the determinisation of Büchi automata, in: FOSSACS'09, in: Lecture Notes in Computer Science, vol. 5504, Springer, 2009, pp. 167–181.

[42] Ph. Schnoebelen, The complexity of temporal logic model checking, in: AIML'02, King's College Publication, 2003, pp. 437–459.

[43] O. Serre, Parity games played on transition graphs of one-counter processes, in: FOSSACS'06, in: Lecture Notes in Computer Science, vol. 3921, Springer, 2006, pp. 337–351.

[44] M. Vardi, P. Wolper, Reasoning about infinite computations, Inform. and Comput. 115 (1994) 1–37.

[45] K.N. Verma, J. Goubault-Larrecq, Karp–Miller trees for a branching extension of VASS, Discrete Math. Theor. Comput. Sci. 7 (2005) 217–230.

[46] S. Vester, On the complexity of model-checking branching and alternating-time temporal logics in one-counter systems, in: ATVA'15, in: Lecture Notes in Computer Science, vol. 9364, Springer, 2015, pp. 361–377.