Strategic Reasoning with a Bounded Number of Resources: the Quest for Tractability

Francesco Belardinelli, Stéphane Demri

Strategic Reasoning with a Bounded Number of Resources: the Quest for Tractability

Francesco Belardinelli

Department of Computing, Imperial College London, UK
and Université Paris-Saclay, Univ Evry, IBISC, 91020, Evry-Courcouronnes, France

Stéphane Demri

Université Paris-Saclay, ENS Paris-Saclay, CNRS, LMF, 91190, Gif-sur-Yvette, France

Abstract

The resource-bounded alternating-time temporal logic RB±ATL combines strategic reasoning with reasoning about resources. Its model-checking problem is known to be 2EXPTIME-complete (the same as its proper extension RB±ATL*) and fragments have been identified to lower the complexity.

In this work, we consider the variant RB±ATL$^+$ that allows for Boolean combinations of path formulae starting with single temporal operators, but restricted to a single resource, providing an interesting trade-off between temporal expressivity and resource analysis. We show that the model-checking problem for RB±ATL$^+$ restricted to a single agent and a single resource is Δ_2^{P}-complete, hence the same as for the standard branching-time temporal logic CTL$^+$. In this case reasoning about resources comes at no extra computational cost. When a fixed finite set of linear-time temporal operators is considered, the model-checking problem drops to PTIME, which includes the special case of RB±ATL restricted to a single agent and a single resource. Furthermore, we show that, with an arbitrary number of agents and a fixed number of resources, the model-checking problem for RB±ATL$^+$ can be solved in EXPTIME using a sophisticated Turing reduction to the parity game problem for alternating vector addition systems with states (AVASS).

1. Introduction

Reasoning about resources with ATL-like logics. Reasoning about multi-agent systems (MAS) – in which autonomous agents can interact and perform actions to reach (joint or individual) goals – has benefited from the model-checking approach with the development of ATL-like logics (see e.g., [1, 2]). In recent years, logic-based languages for the specification and verification of the strategic behaviour of agents in MAS have been the object of increasing interest amongst researchers applying formal methods to artificial intelligence. A wealth of logics for strategies have been proposed in the literature,

Email addresses: francesco.belardinelli@imperial.ac.uk (Francesco Belardinelli), demri@lsv.fr (Stéphane Demri)

including Alternating-Time Temporal Logic ATL [1], possibly extended with strategy contexts [3], Coalition Logic [4], Strategy Logic [5, 6], among others. The expressive power of these formalisms has been thoroughly studied, as well as the corresponding verification problems, thus leading to model-checking tools for concurrent game structures and multi-agent systems [7, 8, 9, 10].

It is worth noticing that the computational models underlying these logic-based languages share a common feature: actions are normally modelled as abstract objects (typically a labelling on transitions) that bear no computational cost. However, if logics for strategies are to be applied to concrete AI scenarios, it is key to account for the resources that actions might consume or produce. These considerations have prompted recently investigations in resource-aware logics for strategies [11, 12]. The need for managing resources in MAS has been identified quite early and many logical formalisms based on ATL have been introduced to endow actions with consumption or production of resources [13, 14, 15, 16, 17]. Unsurprisingly, dealing with resources can lead to high complexity, even undecidable model-checking problems, which is best illustrated in [13, 15, 17]. For instance, the ATL-like logics introduced in [15] are quite rich and general, but it is not obvious which features are to be restricted to regain decidability. Still, decidable resource-aware logics have been identified, for instance by allowing only resource consumption [18]. The challenge for years has been to design resource-aware logics, general enough to allow consumption and production of resources, while having a decidable model-checking problem, possibly low complexity, to allow the implementation in model-checking tools.

From RB±ATL to RB±ATL.* Early decidability results for the model-checking problem for RB±ATL [16, 19] made use of a technique with high complexity (comparable to the complexity of building Karp-Miller trees for Petri nets, see e.g. [20]). The logic RB±ATL is a resource-aware logical formalism that extends ATL by endowing actions with the consumption and production of resources [16]. In [12], its model-checking problem is shown 2EXPTIME-complete by using subroutines to solve decision problems for alternating vector addition systems with states (AVASS), see e.g., [21, 22, 23]. The 2EXPTIME-completeness has also been extended to RB±ATL*, the extension of RB±ATL in which path formulae are arbitrary LTL formulae, in the same way as the temporal logic CTL* extends CTL (see [24, Chapter 7]). The 2EXPTIME upper bound proof for RB±ATL* reveals to be a bit more complex as it requires the determinisation of Büchi automata and a subroutine to solve the parity game problem for AVASS [25], as well as a refined analysis of the complexity function. Consequently, every logic syntactically between RB±ATL and RB±ATL* admits a 2EXPTIME-complete model-checking problem and this also applies to RB±ATL$^+$, the extension of RB±ATL in which path formulae are arbitrary Boolean combinations of LTL formulae of temporal depth one, in the same way as the temporal logic CTL$^+$ extends CTL (see, e.g., [26]). Although the model-checking problem for CTL (resp. CTL$^+$, CTL*) is known to be PTIME-complete (resp. Δ_2^{P}-complete, PSPACE-complete) and the one for ATL (resp. ATL$^+$, ATL*) PTIME-complete [1] (resp. PSPACE-complete [27, 28], 2EXPTIME-complete [1]), there is no difference in complexity for RB±ATL, RB±ATL$^+$ and RB±ATL*, as far as worst-case complexity is concerned.

Our motivation for the present contribution is threefold. Firstly, in the literature there are several gaps in the results available for the decidability and complexity related to the

model-checking problem for RB±ATL [12]. For instance, if we assume two resources and two agents in our multi-agent systems, then the model-checking problem for RB±ATL is known to be PSPACE-hard [29] and in EXPTIME [12, Cor. 1], but no tight complexity result is available. Our long-term aim is to fill all such gaps eventually. Secondly, while completing this picture, it is of interest to identify model-checking instances that are tractable. Although the notion of tractable decision problem is open to discussion, in the context of strategy and temporal logics a model-checking problem decidable in polynomial time (in the size of the formula and the concurrent game structure) falls certainly within the description, as it makes it amenable to implementation in a model-checking tool. Thirdly, complexity results for resource-bounded ATL are proved by using a wealth of different techniques, thus hindering a clear vision of the state of the art. We aim at developing a unified framework based on general proof techniques. Vector addition systems with states (VASS) and their alternating version (AVASS) are key in this respect (see e.g. [20, 21]). Thus, our main motivation consists in reducing the complexity gaps for many meaningful fragments of resource-bounded ATL, in identifying fragments with tractable model-checking, and in proposing optimal algorithms. Finally, we aim at developping proof methods based on counter machines, which happens to be quite natural in view of the nature of the resources.

Let us develop these lines a bit more. When identifying fragments of RB±ATL* of tractable complexity, with the aim of implementing model-checking algorithms in existing tools [8], it may happen that RB±ATL, RB±ATL$^+$, and RB±ATL* have distinct complexity when their fragments are considered, typically by restricting the number of resources or the number of agents. It is commonly accepted that the design of suitable fragments is essential to find a good compromise between the expressive power of the logical language and the computational complexity of the reasoning tasks. That is why the quest for subproblems of RB±ATL* obtained by limiting the temporal expressivity, the class of concurrent game structures, or the number of resources is important for practical use. In this paper, we study specifically the complexity of the model-checking problem for RB±ATL$^+$ (strict syntactic fragment of RB±ATL*) restricted to a bounded number of resources, with a special attention to the case of a single resource. The same approach is also adopted for RB±ATL restricted to a single resource. It is worth noting that as the path formulae in CTL$^+$ (as well as in ATL$^+$ and in RB±ATL$^+$) are Boolean combinations of LTL formulae of temporal depth one, this provides a generic and natural way to design linear-time constraints. Surprisingly, this specific viewpoint considering fragments of RB±ATL$^+$ happens to be fruitful, not only for the identification of interesting fragments of lower complexity, but also in terms of proof techniques to solve the model-checking problem (see Section 4).

Our contribution. Based on the logics RB±ATL and RB±ATL* studied in [12] and for which the respective model-checking problems are 2EXPTIME-complete, we identify tractable fragments and lower the worst-case complexity for other fragments. Details follow.

1. We introduce the logic RB±ATL$^+$, similarly to the way ATL$^+$ (resp. CTL$^+$) extends ATL (resp. CTL). We prove that the model-checking problem for RB±ATL$^+$ restricted to a single agent and to a single resource is Δ_2^{P}-complete (Theorem 11). The Δ_2^{P} upper bound combines ingredients for model-checking CTL$^+$ with procedures in PTIME for decision problems for vector addition systems with states

(VASS) [20] restricted to a single counter, newly introduced in this paper. Since CTL^+ model-checking is also Δ_2^{P}-complete [30], we observe that the enhanced expressivity provided by reasoning about a single resource comes at no extra computational cost.

2. Related to point 1, we introduce the new generalised control-state reachability problem for VASS and show it to be in PTIME when restricted to VASS with one counter (1-VASS) (Theorem 10). The same problem with an unrestricted number of counters can be easily shown to be EXPSPACE-complete, but its PTIME subproblem with one counter is instrumental to get the above-mentioned Δ_2^{P}-completeness result. This PTIME upper bound for the generalised control-state reachability problem for 1-VASS is partly based on the complexity results for the state-reachability and nontermination problems in VASS with a single counter. As regards the latter problems, [31] handles the boundedness problem in a way that can be adapted for the nontermination problem and [32, Section 3.3] actually handles state-reachability for 1-VASS. In Section 3.2, we provide a uniform treatment (see Theorems 7 and 8) while introducting a few minor necessary updates to the proof of [31, Theorem 3.4]. Hence, we believe that this part of the paper can be considered as self-standing contributions allowing us to design a fine-tuned algorithm to solve the generalised control-state reachability problem restricted to 1-VASS.

3. We prove that model-checking $\mathsf{RB}{\pm}\mathsf{ATL}$ is PTIME-complete when we reason about a single resource and a single agent. In the paper, this is done in two ways. First, we show the PTIME upper bound when the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+$ is restricted to a single resource, a single agent, and a finite set of linear-time operators (Theorem 15). $\mathsf{RB}{\pm}\mathsf{ATL}$ is then just a special case. The second method shows the PTIME upper bound directly by using polynomial-time algorithms for the state-reachability and nontermination problems in VASS with a single counter as well as the PTIME lower bound from CTL model-checking (see Section 5.2). Since we show that this setting is tantamount to the Computation Tree Logic CTL with a single resource (Theorem 19), our result means that we can reason about a single resource in CTL too at no extra computational cost.

4. Furthermore, we generalise point 1 by showing that the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+$ restricted to r resources (for a fixed r) can be solved in EXPTIME (Theorem 13) by reduction to the parity game problem for AVASS [25]. Remarkably, the number of locations in the target AVASS is exponential in the input size, which makes a substantial difference w.r.t. what is done in [12] for $\mathsf{RB}{\pm}\mathsf{ATL}^*$, where the number of locations is double exponential. Hence, the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+$ restricted to one resource is PSPACE-hard (inherited from ATL^+ [27, 28]) and in EXPTIME. Apart from the complexity results that improve our understanding of fragments of $\mathsf{RB}{\pm}\mathsf{ATL}^*$, we present reductions to the parity game problem for AVASS that can be of interest for their own sake.

Related Work. Resource-aware formalisms have a well-established tradition in mathematical logic, dating back at least to the work on substructural logics [33, 34, 35]. Hereafter we focus specifically on the several ATL-like formalisms that have been put forward in recent years, which are characterized by endowing actions with the consumption

or production of resources [11, 13, 14, 15, 16, 17]. In this line of research, a remarkable breakthrough occurred with the design of the logic RB±ATL, with production (+) and consumption (−) of resources, whose model-checking problem was shown decidable in [16]. Subsequently, fragments of RB±ATL have been designed with relatively low complexity (see, e.g., [36, 19]).

In the same line, [37] considers Resource Agent Logic (RAL), which extends ATL to verify properties of systems where agents act under resource constraints. Unlike RB±ATL, in RAL the agents do not necessarily re-equip their resources whenever new strategies are considered and resources are allowed to be updated based on actions taken by agents from the opponent coalition. In [37], the authors review existing (un)decidability results for fragments of RAL with unbounded production and consumption of resources, tighten some existing undecidability results, and identify several aspects which affect decidability of model-checking, including the availability of a 'do nothing', or idle action.

Similary, in [12] the authors revisit decidability results for resource-aware logics, with the notable difference of leveraging on decision problems for vector addition systems with states (VASS), in order to establish complexity characterisations of (decidable) model-checking problems. In particular, they show that the model-checking problem for the logic RB±ATL is 2EXPTIME-complete by using recent results on alternating VASS [21, 25]. We refer to Tables 2–4 in Section 2 for a detailed list of the complexity results proved therein. Similarly to [12], in the present contribution we still leverage on VASS, but differently from [12, 37] we provide complexity results also for the extension RB±ATL$^+$ of RB±ATL. Moreover, our results do not rely on the availability of an idle action, differently from [37].

Finally, in [36] the authors give a symbolic model-checking algorithm for the 1RB±ATL fragment of RB±ATL that allows only one resource type. They evaluate the performance of an implementation of the algorithm on an example problem that can be scaled to large state spaces. Here, we do not tackle the problem of efficient implementation of the decision procedures we provide.

For a thorough discussion of the current state of the art in resource-bounded ATL, we refer to the survey paper [11].

Structure of the paper. The rest of the paper is structured as follows. In Section 2 we present the syntax and semantics of resource-bounded alternating-time temporal logic. In particular, we define the logic RB±ATL* and its fragments RB±ATL$^+$ and RB±ATL, we outline the state of the art on their model-checking problems, as well as the novel results we prove. In Section 3 we introduce some background notions on vector addition systems with states (VASS), specifically for the case of a single counter. For these 1-VASS, we define the nontermination and generalised reachability problems, which are used to provide a unified framework to solve the model-checking problem for RB±ATL$^+$ in Section 4, where we first consider the case of a single agent and a single resource (Section 4.1), and then the case for a fixed number r of resources (Section 4.2). Finally, in Section 5 we present some notable extensions of our results, specifically to a bounded number of temporal operators (Section 5.1) and to the fragment RB±ATL (Section 5.2). In Section 5.3, we provide a comparison with the logic RBTL from [13, 38]. In short, RBTL can be understood as the variant of RB±ATL using only the full set of agents or the empty set as coalitions. We conclude in Section 6 by pointing to future lines of work.

Previous work by the same authors. This paper is a complete and extended version of

the conference articles [39] and [40] by the same authors. However, we added several original contributions. In particular, the results pertaining to point 3 above, about the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+$ restricted to a single resource, a single agent, and a fixed finite set of linear-time operators, are all new and did not appear in [39, 40]. Furthermore, the paper presents full proofs and extensive discussion of results and proof techniques, pointing to possible future extensions and applications.

2. Resource-Bounded Alternating-time Temporal Logics

The presentation below for resource-bounded alternating-time temporal logics follows closely [12, 39, 40]. In the rest of the paper, $\mathbb{N}$ (resp. $\mathbb{Z}$) is the set of natural numbers (resp. integers) and for $m, m' \in \mathbb{Z}$, $[m, m']$ is the set $\{j \in \mathbb{Z} \mid m \leq j \leq m'\}$. For a finite or infinite sequence $w \in X^* \cup X^\omega$ of elements in some given set X, we write w_i or $w[i]$ for the $(i+1)$-th element of w, i.e., $w = w_0 w_1 \ldots$. For $i \geq 0$, $w_{\leq i} = w_0 w_1 \ldots w_i$ is the prefix of w of length $i+1$, while $w_{\geq i} = w_i w_{i+1} \ldots$ is the suffix of w starting at position i. The length of a (finite or infinite) sequence $w \in X^\omega \cup X^*$ is denoted as $|w|$, where $|w| = \omega$ for $w \in X^\omega$.

2.1. The logic RB±ATL and its fragments*

To specify the strategic properties of agents in resource-bounded multi-agent systems, we present the logic $\mathsf{RB}{\pm}\mathsf{ATL}^*$ as well as its fragments $\mathsf{RB}{\pm}\mathsf{ATL}^+$ and $\mathsf{RB}{\pm}\mathsf{ATL}$, which are resource-aware extensions of the alternating-time temporal logics ATL^*, ATL^+, and ATL respectively, introduced in [16, 41] to explicitly account for the production and consumption of resources by agents. In the rest of the paper, we assume a countably infinite set AP of atomic propositions (or atoms). Given a finite non-empty set Ag of agents and a number $r \geq 1$ of resources (often called "resource types" in the literature), we write $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ to denote the resource-bounded logic with agents from set Ag and r resources.

Syntax. The models of $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ and its fragments are resource-bounded concurrent game structures (see a formal introduction in Definition 3) in which transitions are labelled by tuples of actions of length $|Ag|$, each agent performing one action. A notion of computation arises naturally as an infinite sequence of consecutive states.

Definition 1 ($\mathsf{RB}{\pm}\mathsf{ATL}^*$). *The* state formulae ϕ *and* path formulae Ψ *in the logic* $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ *are built according to the following BNF grammar:*

$$\begin{array}{lcl} \phi & ::= & p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle A^{\vec{b}} \rangle\rangle \Psi \\ \Psi & ::= & \phi \mid \neg\Psi \mid \Psi \wedge \Psi \mid \mathsf{X}\,\Psi \mid \Psi\,\mathsf{U}\,\Psi \end{array}$$

where $p \in \mathrm{AP}$*,* $A \subseteq Ag$*, and* $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$*.*

By $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ formulae*, by default we understand all and only the state formulae.*

Similarly to the temporal logic CTL^*, the syntax of $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ distinguishes state formulae (interpreted on states) from path formulae (interpreted on computations). The main feature of $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ are state formulae of the form $\langle\langle A^{\vec{b}} \rangle\rangle \Psi$,

where $A \subseteq Ag$ is a *coalition* (set of agents), Ψ is a path formula defining a property on computations, and $\vec{b}$ is an initial budget, i.e., a r-tuple over $\mathbb{N} \cup \{\omega\}$, understood as a quantity for each resource type, where the value ω accounts for an infinite supply of the respective resource. Roughly speaking, $\langle\langle A^{\vec{b}} \rangle\rangle \Psi$ is read as "the coalition A of agents has a joint strategy implementable with budget $\vec{b}$ such that all the computations respecting that strategy satisfy the linear-time property Ψ". The initial budget is relevant as in resource-bounded concurrent game structures, actions come with resource consumption and production. In particular, the operator $\langle\langle A^{\vec{b}} \rangle\rangle$ quantifies over all computations such that the resource counters are non-negative provided the initial budget and the cost of actions in the relevant strategy. It is worth noting that a formula $\langle\langle A^{\vec{b}} \rangle\rangle \Psi$ is interpreted as a property that alternates one existential quantification (over strategies) and one universal quantification (over computations), similarly to what is done in ATL [1].

The linear-time operators *next* X and *until* U from LTL have their standard readings; the propositional connectives $\vee$, $\Rightarrow$, and the temporal operators *eventually* F and *always* G are introduced as usual. The dual operator $[\![A^{\vec{b}}]\!]$ is introduced as $[\![A^{\vec{b}}]\!]\Psi \stackrel{\text{def}}{=} \neg\langle\langle A^{\vec{b}} \rangle\rangle \neg\Psi$ (in $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$, path formulae are closed under negation, but this does not hold for all its fragments).

Now let us introduce the fragments $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$ and $\mathsf{RB}{\pm}\mathsf{ATL}(Ag, r)$ that are, along the lines of the definition of CTL^+ and CTL from CTL^*.

Definition 2 ($\mathsf{RB}{\pm}\mathsf{ATL}^+$ and $\mathsf{RB}{\pm}\mathsf{ATL}$). *State formulae in $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$ are obtained from the BNF grammar for $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ by restricting path formulae as follows:*

$$\Psi \quad ::= \quad \neg\Psi \mid \Psi \wedge \Psi \mid \mathsf{X}\,\phi \mid \phi\,\mathsf{U}\,\phi$$

Similarly, state formulae in $\mathsf{RB}{\pm}\mathsf{ATL}(Ag, r)$ are obtained from the BNF grammar for the logic $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ by restricting path formulae as follows:

$$\Psi \quad ::= \quad \mathsf{X}\,\phi \mid \phi\,\mathsf{U}\,\phi \mid \mathsf{G}\,\phi$$

Notice that $\mathsf{G}\,\phi$ is usually introduced as a shortcut for $\neg(\top\,\mathsf{U}\,\neg\phi)$ in $\mathsf{RB}{\pm}\mathsf{ATL}^+$, but it has to be assumed as a primitive symbol in $\mathsf{RB}{\pm}\mathsf{ATL}$.

The logics $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$ and $\mathsf{RB}{\pm}\mathsf{ATL}(Ag, r)$ introduce increasing restrictions on the form of path formulae Ψ as goals in the scope of strategic operators. In $\mathsf{RB}{\pm}\mathsf{ATL}(Ag, r)$ only simple goals of the form $\mathsf{X}\,\phi$, $\phi\,\mathsf{U}\,\phi$, and $\mathsf{G}\,\phi$ are allowed. In $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$, we can also have Boolean combinations thereof. Finally, $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$ allows any combination of linear-time operators as temporal goals. We will see that the increased expressivity has an impact on the complexity of the corresponding model-checking problems. We write $\mathsf{RB}{\pm}\mathsf{ATL}^*(r)$ to denote the set of formulae in $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$, for some fixed non-empty set Ag of agents. Similar definitions are used for $\mathsf{RB}{\pm}\mathsf{ATL}^+(r)$ and $\mathsf{RB}{\pm}\mathsf{ATL}(r)$.

Semantics. We interpret the formulae in $\mathsf{RB}{\pm}\mathsf{ATL}^*$ (and therefore $\mathsf{RB}{\pm}\mathsf{ATL}^+$ and $\mathsf{RB}{\pm}\mathsf{ATL}$) by using resource-bounded concurrent game structures [42, 12], i.e., concurrent game structures endowed with a weight function that assigns an integer to every action (hereafter understood as a gain).

Definition 3 (RB-CGS). *A* resource-bounded concurrent game structure *is a tuple* $M = \langle Ag, r, S, Act, act, wf, \delta, L\rangle$ *such that*

- Ag *is the finite, non-empty set of agents (by default* $Ag = [1, k]$ *for some* $k \geq 1$*);*
- $r \geq 1$ *is the number of* resources*;*
- S *is a finite, non-empty set of* states*;*
- Act *is a finite, non-empty set of* actions*;*
- $act : S \times Ag \to (\wp(Act) \setminus \emptyset)$ *is the* protocol function*;*
- $wf : S \times Ag \times Act \to \mathbb{Z}^r$ *is the (partial)* weight function*; that is, for* $s \in S$*,* $a \in Ag$*, and* $\mathtt{a} \in Act$*,* $wf(s, a, \mathtt{a})$ *is defined only when* $\mathtt{a} \in act(s, a)$*;*
- $\delta : S \times (Ag \to Act) \to S$ *is the (partial)* transition function *such that* δ *is defined for a state* s *and a joint action* $f : Ag \to Act$ *only if for every agent* $a \in Ag$*,* $f(a) \in act(s, a)$*;*
- $L : \mathrm{AP} \to \wp(S)$ *is the* (partial) labelling function.

Intuitively, a resource-bounded CGS describes the interactions of a group Ag of agents, who perform the actions in Act according to the protocol function act. The execution of a joint action $f : Ag \to Act$ generates a transition in the system, as specified by the transition function δ. Moreover, on each transition the values of the r resources are updated according to the weights wf of the joint action. Definition 3 excludes the case with no resource types ($r = 0$) because in this paper we emphasize the use of resources. However, by convention, having zero resource types in RB-CGS corresponds to the standard concurrent game structures from [1]. It is worth noting that in Definition 3 we do not assume the existence of an `idle` action (with zero weight), which was first introduced in [16, 19] and it is often advantageous in terms of computational complexity (see, e.g. [37, 12]). In this paper we do not rely on the action `idle` to establish our complexity results, but of course, nothing prevents us from having such a distinguished action in our instances of the model-checking problem. The only exception for which we require the existence of the idle action is in Section 5.2. Indeed, the existence of such an action with no effect on resources allows us to extend computations from any state, which is helpful for solving control-state reachability problems.

An RB-CGS M is *finite* whenever L is restricted to a finite subset of AP. The size $|M|$ of a finite RB-CGS M is the size of its encoding when integers are encoded in binary and maps and sets are encoded in extension using a reasonably succinct encoding. All the complexity results in the paper are about upper bounds. Using a binary encoding for integers, which is the common practice in computer science, usually requires a bit more efforts than a unary encoding. In this work, conciseness of the binary encoding for integers does not lead to substantial difficulties when proving complexity upper bounds.

Now, we provide the necessary notions to interpret our languages on resource-bounded CGS. Given a coalition $A \subseteq Ag$ and a state $s \in S$, a *joint action available to* A *in* s is a map $f : A \to Act$ such that for every agent $a \in A$, $f(a) \in act(s, a)$. The set of all such joint actions is denoted as $D_A(s)$. Given joint actions f, g, we write $f \sqsubseteq g$ if

$Dom(f) \subseteq Dom(g)$, and for every agent $a \in Dom(f)$, $g(a) = f(a)$. For a joint action $f \in D_A(s)$, $\mathtt{out}(s, f)$ is the set of *immediate outcomes*:

$$\mathtt{out}(s, f) \stackrel{\text{def}}{=} \{\delta(s, g) \mid \text{ for some } g \in D_{Ag}(s), f \sqsubseteq g\}$$

Further, the weight of a transition from s by f (w.r.t. coalition A) is defined as

$$wf_A(s, f) \stackrel{\text{def}}{=} \sum_{a \in A} wf(s, a, f(a))$$

A *computation* λ is a finite or infinite sequence $s_0 \stackrel{f_0}{\to} s_1 \stackrel{f_1}{\to} s_2 \ldots$ such that for all $0 \leq i < |\lambda| - 1$ we have $s_{i+1} = \delta(s_i, f_i)$.

To provide a formal semantics to the strategic operator $\langle\langle A^{\vec{b}} \rangle\rangle$ based on RB-CGS, we need a notion of resource-bounded strategy.

Definition 4 (Strategy). *A* (memoryful) 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* $F_A(s_0 \xrightarrow{f_0} s_1 \ldots s_{n-1} \xrightarrow{f_{n-1}} s_n) \in D_A(s_n)$.

A computation $\lambda = s_0 \stackrel{f_0}{\to} s_1 \stackrel{f_1}{\to} s_2 \ldots$ respects the strategy F_A *iff for all* $i < |\lambda| - 1$, $s_{i+1} \in \mathtt{out}(s_i, F_A(s_0 \stackrel{f_0}{\to} s_1 \ldots s_{n-1} \xrightarrow{f_{i-1}} s_i))$.

A computation λ that respects F_A is *maximal* if it cannot be extended further while respecting F_A. Therefore, maximal computations starting in state s and respecting F_A are infinite, and we denote the set of all such maximal computations by $\mathtt{Comp}(s, F_A)$. Given a bound $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$ and a computation $\lambda = s_0 \stackrel{f_0}{\to} s_1 \stackrel{f_1}{\to} s_2 \ldots$ in $\mathtt{Comp}(s, F_A)$, let the *resource availability* $\vec{v}_i$ *at step* $i \in \mathbb{N}$ be defined as: $\vec{v}_0 = \vec{b}$ and for $i \geq 0$, $\vec{v}_{i+1} = wf_A(s_i, f_i) + \vec{v}_i$ (assuming $n + \omega = \omega$ for every $n \in \mathbb{Z}$). Then, λ is $\vec{b}$*-consistent* iff for all $i \in \mathbb{N}$, we have $\vec{v}_i \in (\mathbb{N} \cup \{\omega\})^r$. A $\vec{b}$*-strategy* F_A *with respect to* s is a strategy such that all the maximal computations in $\mathtt{Comp}(s, F_A)$ are $\vec{b}$-consistent. In particular, this means that resource quantities are non-negative along the corresponding computations. If $\vec{b}[i] = \omega$, we actually have an infinite supply of the i-th resource, thus not constraining the behaviour of agents with respect to that particular resource and effectively disregarding what happens about it. Since the resource availability depends only on the agents in coalition A, this is called the *proponent restriction condition* (see, e.g., [17, 12]). This asymmetry between the proponent and the opponent coalitions is particularly helpful to guarantee decidability of the model-checking problem, though resources can be produced or consumed by the proponent coalition. The introduction of such a condition in [17] is strongly motivated with the goal of getting decidable model-checking problems.

Definition 5 (Satisfaction). *Given a state* $s \in S$*, an infinite computation* λ*, an atom* $p \in \mathrm{AP}$*, a state formula* ϕ *and a path formula* Ψ *in* RB±ATL**, the satisfaction relation* $\models$ *is defined as follows:*

$(M, s) \models p$ *iff* $s \in L(p)$
$(M, s) \models \neg\phi$ *iff* $(M, s) \not\models \phi$
$(M, s) \models \phi_1 \wedge \phi_2$ *iff* $(M, s) \models \phi_i$ *for* $i \in \{1, 2\}$

$(M,s) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$ *iff* *for some $\vec{b}$-strategy F_A w.r.t. s, for all computations $\lambda \in \mathtt{Comp}(s, F_A)$, $(M,\lambda) \models \Psi$*

$(M,\lambda) \models \phi$ *iff* $(M, \lambda[0]) \models \phi$

$(M,\lambda) \models \neg\Psi$ *iff* $(M,\lambda) \not\models \Psi$

$(M,\lambda) \models \Psi_1 \wedge \Psi_2$ *iff* $(M,\lambda) \models \Psi_i$ *for* $i \in \{1,2\}$

$(M,\lambda) \models \mathsf{X}\,\Psi$ *iff* $(M, \lambda_{\geq 1}) \models \Psi$

$(M,\lambda) \models \Psi\,\mathsf{U}\,\Psi'$ *iff* *for some $i \geq 0$, $(M,\lambda_{\geq i}) \models \Psi'$, and for all $0 \leq j < i$, $(M,\lambda_{\geq j}) \models \Psi$*

Clearly, the alternating-time temporal logics ATL^*, ATL^+, and ATL can be seen as syntactic fragments of $\mathsf{RB}{\pm}\mathsf{ATL}^*$, $\mathsf{RB}{\pm}\mathsf{ATL}^+$, and $\mathsf{RB}{\pm}\mathsf{ATL}$ respectively. In particular, the unindexed strategic operator $\langle\langle A \rangle\rangle$ can be expressed as $\langle\langle A^{\vec{\omega}} \rangle\rangle$. In order to stick to the definition of the original version of $\mathsf{RB}{\pm}\mathsf{ATL}$ in [16] (see also [12]) – as well as to provide more flexibility to the specification language – we keep the bound ω, but the way to handle it consists in reducing the dimension (i.e., eliminating the components where it appears, as it does not bring any quantitative constraint), which can be easily performed.

Remark. In the case of a single agent, that is, for $Ag = \{1\}$, in our languages we only have modalities $\langle\langle Ag^{\vec{b}} \rangle\rangle$ and $\langle\langle \emptyset^{\vec{b}} \rangle\rangle$, as well as duals $[\![Ag^{\vec{b}}]\!]$ and $[\![\emptyset^{\vec{b}}]\!]$, for $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$.

By Definition 5, the meaning of these operators is as follows:

$(M,s) \models \langle\langle \emptyset^{\vec{b}} \rangle\rangle \Psi$ iff for every computation λ from s, $(M,\lambda) \models \Psi$

$(M,s) \models [\![\emptyset^{\vec{b}}]\!] \Psi$ iff for some computation λ from s, $(M,\lambda) \models \Psi$

$(M,s) \models \langle\langle Ag^{\vec{b}} \rangle\rangle \Psi$ iff for some $\vec{b}$-consistent computation λ from s, $(M,\lambda) \models \Psi$

$(M,s) \models [\![Ag^{\vec{b}}]\!] \Psi$ iff for every $\vec{b}$-consistent computation λ from s, $(M,\lambda) \models \Psi$

Notice that the semantics of operators $\langle\langle \emptyset^{\vec{b}} \rangle\rangle$ and $[\![\emptyset^{\vec{b}}]\!]$ corresponds to the meaning of the universal (A) and existential (E) path quantifiers in CTL^*, where the weights in the RB-CGS are ignored because of the proponent restriction condition. Hence, $\mathsf{RB}{\pm}\mathsf{ATL}^*(\{1\}, r)$ can be thought of as a resource-bounded version of CTL^* with r resources. In Section 5.3, we show that $\mathsf{RB}{\pm}\mathsf{ATL}^*$ for the single agent case is basically equivalent to a different resource-bounded logic RBTL^* introduced in [38]. ■

We illustrate the formal machinery introduced so far, particularly the single-agent case, with two AI-oriented examples.

Example 1. We consider a scenario in which a rover is exploring an unknown area. At any time the rover can choose between two modes: either it moves around or it recharges its battery through a solar panel, but it cannot do both things at the same time. Moving around consumes one energy unit at every time step, whereas the rover can recharge of one energy unit at a time. Switching between these modes also requires one energy unit.

This simple scenario can be modelled as the resource-bounded CGS

$$M = \langle \{rover\}, 1, \{s_1, s_2\}, \{\mathtt{move}, \mathtt{recharge}, \mathtt{switch}\}, act, wf, \delta, L \rangle$$

depicted in Fig. 1, where in particular:

- $act(s_1, rover) = \{\mathtt{move}, \mathtt{switch}\}$ and $act(s_2, rover) = \{\mathtt{recharge}, \mathtt{switch}\}$;

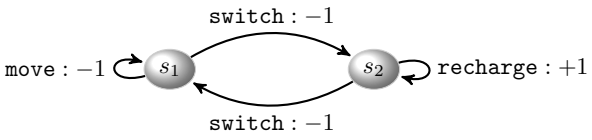


Figure 1: the resource-bounded CGS in Example 1.

- $wf(s_1, rover, \texttt{move}) = wf(s_1, rover, \texttt{switch}) = wf(s_2,\ rover, \texttt{switch}) = -1$ and $wf(s_2, rover, \texttt{recharge}) = +1$;
- $\delta(s_1, \texttt{move}) = s_1$, $\delta(s_1, \texttt{switch}) = s_2$, $\delta(s_2, \texttt{recharge}) = s_2$, and $\delta(s_2, \texttt{switch}) = s_1$;
- $AP = \{moving\}$ and $L(moving) = \{s_1\}$.

Even in such a simple scenario with a single agent, we can express interesting properties such as "no matter what the rover does, at any time it has a strategy, with an initial budget of b energy units, such that it will eventually be moving". This specification can be expressed as

$$[\![\{rover\}^\omega]\!]\, \mathsf{G}(\langle\!\langle \{rover\}^b \rangle\!\rangle\, \mathsf{F}\, moving) \tag{1}$$

It is worth noting that formula (1) is not in RB±ATL but the next formula is. I.e., it is easy to check that (1) above is logically equivalent to the following formula:

$$\neg\langle\!\langle \{rover\}^\omega \rangle\!\rangle\, \mathsf{F}\, \neg(\langle\!\langle \{rover\}^b \rangle\!\rangle\, \mathsf{F}\, moving).$$

In Section 5.1 and 5.2 we show that specifications such as (1), concerning a single agent and a single resource, can be efficiently verified in PTIME. ⊣

Here is another example, tailored to illustrate the expressive power of RB±ATL$^+$ formulae.

Example 2. We consider a scenario in which two drones, d_1 and d_2, have to deliver two parcels to addresses (X_1, Y_1) and (X_2, Y_2) on a $[-m, m]^2$ grid for some $m > 0$. We assume that they both start from location $(0, 0)$. At any time, each drone can choose between two modes: either it moves in one of the four directions (N, S, E, W) or it recharges its battery through a solar panel (charge), but it cannot do both actions at the same time. Moving around consumes one energy unit at every time step; whereas the drone can recharge one unit at a time. Each drone can carry one parcel at a time, and they can only pick them up from location $(0, 0)$.

This scenario can be modelled as a resource-bounded CGS

$$M = \langle \{d_1, d_2\}, 1, [-m, m]^2 \times [-m, m]^2, \{\texttt{N}, \texttt{S}, \texttt{E}, \texttt{W}, \texttt{charge}\}, act, wf, \delta, L \rangle$$

where in every state s, for $i \in \{1, 2\}$ and $\texttt{D}_i \in \{\texttt{N}, \texttt{S}, \texttt{E}, \texttt{W}\}$:

- $act(s, d_i) = \{\texttt{N}, \texttt{S}, \texttt{E}, \texttt{W}, \texttt{charge}\}$;
- $wf(s, d_i, \texttt{D}_i) = -1$ and $wf(s, d_i, \texttt{charge}) = +1$;

- $\delta(((x_1, y_1), (x_2, y_2)), (\mathtt{D}_1, \mathtt{D}_2)) = ((x'_1, y'_1), (x'_2, y'_2))$, where (x'_i, y'_i) is obtained from (x_i, y_i) by decrementing/incrementing either x_i or y_i depending on $\mathtt{D}_i$. Notice that on the frontier, movement in some directions is assumed to have no effect.

  Similarly for $\delta(((x_1, y_1), (x_2, y_2)), (\mathtt{D}_1, \mathtt{charge}))$ and $\delta(((x_1, y_1), (x_2, y_2)), (\mathtt{charge}, \mathtt{D}_2))$.

  Moreover, $\delta(s, (\mathtt{charge}, \mathtt{charge})) = s$.

- AP $= \{charge_1, charge_2, del_1, del_2\}$ and $L(del_i) = \{(X_i, Y_i)\}$, while a state is labelled with $charge_i$ if drone d_i has charged as previous action (we can keep track of this information with an extra bit of memory in the system state).

Even in such a simple scenario we can express interesting strategic properties in $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{d_1, d_2\}, 1)$. Consider a budget $b = \max\{(|X_1| + |Y_1|), (|X_2| + |Y_2|)\}$ that allows any drone d to deliver a single parcel to the farthest address. We can check that in location $((0,0),(0,0))$ the following formulae are true:

1. With budget b, every drone d_i, for $i \in \{1, 2\}$, can deliver parcel 1 and can deliver parcel 2, without charging in the meantime:
$$\langle\!\langle \{d_i\}^b \rangle\!\rangle (\neg charge \,\mathsf{U}\, del_1) \wedge \langle\!\langle \{d_i\}^b \rangle\!\rangle (\neg charge \,\mathsf{U}\, del_2) \tag{2}$$
2. With budget $2b$, neither drone can deliver both parcel 1 and 2, without charging in the meantime (recall that drones have to pick parcels in location $(0,0)$):
$$\neg\langle\!\langle \{d_i\}^{2b} \rangle\!\rangle ((\neg charge \,\mathsf{U}\, del_1) \wedge (\neg charge \,\mathsf{U}\, del_2)) \tag{3}$$
3. Nonetheless, with budget $2b$, the drones can cooperate to deliver both parcels, without charging in the meantime:
$$\langle\!\langle \{d_1, d_2\}^{2b} \rangle\!\rangle ((\neg charge \,\mathsf{U}\, del_1) \wedge (\neg charge \,\mathsf{U}\, del_2)) \tag{4}$$

Notice that formula (2) belongs to $\mathsf{RB}{\pm}\mathsf{ATL}$, but we make use of Boolean combinations of goals and the expressivity of $\mathsf{RB}{\pm}\mathsf{ATL}^+$ to formalize (3) and (4). ⊣

2.2. The model-checking problem: state of the art

Hereafter $\mathcal{L}$ is a (possibly trivial) fragment of $\mathsf{RB}{\pm}\mathsf{ATL}^*$, typically $\mathsf{RB}{\pm}\mathsf{ATL}$, $\mathsf{RB}{\pm}\mathsf{ATL}^+$ or $\mathsf{RB}{\pm}\mathsf{ATL}^*$, or some of its restrictions. In the sequel, we consider the following decision problem.

Definition 6 (Model-checking). *Let M be a finite RB-CGS with $Ag = [1, k]$ and r resources for some $k, r \geq 1$, and let s be a state in M. Let ϕ be a formula in a logic $\mathcal{L}$ built over Ag and r resources. The model-checking problem amounts to decide whether $(M, s) \models \phi$.*

Given a logic $\mathcal{L}$, we write MC($\mathcal{L}$) to denote the corresponding model-checking problem. Below, we summarize known complexity results for subproblems of MC($\mathcal{L}$), when $\mathcal{L}$ is either $\mathsf{RB}{\pm}\mathsf{ATL}^*$, $\mathsf{RB}{\pm}\mathsf{ATL}^+$, or $\mathsf{RB}{\pm}\mathsf{ATL}$, obtained by bounding the number k of agents and r of resources. We also identify the contributions made in this paper.

$r \backslash \lvert Ag \rvert$	∞	$\geq \mathbf{2}$	$\mathbf{1}$
∞	2EXPTIME-c. [12, Th. 2 and 3]		EXPSPACE-c. [12, Th. 4]
$\geq \mathbf{4}$	EXPTIME-c. [12, Cor. 1] [21] (†)		in PSPACE [12, Cor. 2] PSPACE-h. [29]
3	in EXPTIME [12, Cor. 1] (†) PSPACE-h. [29]		
2			
1	in PSPACE [19, Th. 2] (†) PTIME-h. (from ATL)		in PTIME **(Cor. 2)** PTIME-h. (from CTL)

Figure 2: Complexity of model-checking RB±ATL(Ag, r). Results proved in this paper are in boldface.

2.2.1. Complexity of MC(*RB±ATL*) *and fragments*

In Figure 2, we summarize the main complexity results available in the literature for RB±ATL(Ag, r). The PTIME upper bound in boldface was originally presented by the authors in [39], herein we provide the full proofs (see Sections 4.2, 5.2 and 5.2). It is worth noting that all the results are known to hold in the presence of the release operator R instead of the always operator G, except for the PSPACE upper bound from [19, Th. 2] (for which further investigations would be needed). The symbol '∞' corresponds to the case when either r or $|Ag|$ is unbounded. By contrast, '$\geq n$' means that the number of either resources or agents is fixed, but greater than n. Moreover, some of the results assume the existence of an idle action (roughly, it has no effect on resources, but it can be choosen anytime by any agent). This assumption does not impact on the lower bounds, but it can make a difference for the upper bounds. We write '(†)' to indicate that the corresponding result is proved under the assumption of the idle action in the cited work (though further investigations might show that even without the idle action, the complexity upper bound is preserved).

We now discuss briefly the main results in Figure 2. For an unbounded number of resources and at least two agents, the model-checking problem is known to be 2EXPTIME-complete. This result follows from Theorem 2 (upper bound) and Theorem 3 (lower bound) in [12]. When restricted to a single agent, the same problem becomes EXPSPACE-complete [12, Th. 4].

For a fixed number of resources greater or equal than four and at least two agents, the model-checking problem is EXPTIME-complete. The upper bound follows from [12, Cor. 1], while the lower bound derives from the complexity of the control-state reachability problem for alternating VASS [21], which can be simulated by using at least two agents only [12, Th. 3]. Further, for a fixed amount of resources greater or equal than two, and at least two agents, the model-checking problem is in EXPTIME [12, Cor. 1]. In the case of a single agent, the same problem is in PSPACE [12, Cor. 2]. Moreover, it is PSPACE-hard in both cases, as we can reduce to it the control-state reachability problem for 2-VASS, which is PSPACE-complete [29]. It is worth noting that in [12], all the RB-CGS are assumed to admit an idle action.

Finally, in the case of a single resource, the problem is known to be in PSPACE in general [19, Th. 2]. We observe that the case of a single resource can also capture situations in which $r > 1$ resources can be converted into a unique resource (e.g., money), possibly with different rates. Under the extra restriction of a single agent, model-checking is in PTIME, which is our theoretical contribution in Section 5.1 based on results in

$r\backslash\|Ag\|$	∞	$\geq \mathbf{2}$	**1**
∞	2EXPTIME-c. [12, Th. 2 and 3]		EXPSPACE-c. [12, Th. 4]
$\geq \mathbf{4}$	in EXPTIME **(Th. 13)** EXPTIME-h. [12, Th. 2 and 3]		PSPACE-c [12, Cor. 2] [27]
3	in EXPTIME **(Th. 13)** PSPACE-h. (from ATL^+ [27])		
2			
1			in Δ_2^{P} **(Th. 11)** Δ_2^{P}-h. (from CTL^+ [30])

Figure 3: The complexity of model-checking $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$. Results proved in this paper are in boldface.

$r\backslash\|Ag\|$	∞	$\geq \mathbf{2}$	**1**
∞	in 2EXPTIME [12, Th. 7] 2EXPTIME-h. (from ATL^*)		EXPSPACE-c. [12, Th. 8]
$\geq \mathbf{1}$			in PSPACE ([12, Cor. 2] & **Th. 19**) PSPACE-h. (from CTL^*)

Figure 4: The complexity of model-checking $\mathsf{RB}{\pm}\mathsf{ATL}^*(Ag, r)$. Results proved in this paper are in boldface.

Section 4.2 (see also Section 5.2).

It is therefore PTIME-complete as the CTL model-checking problem is already PTIME-hard (see, e.g. [43, 24]). The characterisation of the complexity for one resource and at least two agents is still open: currently, neither the proof of the PTIME upper bound in Section 5.2, nor the PSPACE-hardness results from [44] and [45, Sect. 5] could be advantageously used to close this complexity gap.

2.2.2. Complexity of MC($RB{\pm}ATL^+$) *and fragments*

As regards $\mathsf{RB}{\pm}\mathsf{ATL}^+$ (see Figure 3), for an unbounded number of resources we have the same complexity results as for $\mathsf{RB}{\pm}\mathsf{ATL}$: 2EXPTIME-completeness for at least two agents, and EXPSPACE-completeness in the single-agent case. Also, for a single agent, the problem is PSPACE-complete when considering at least two resources, just like $\mathsf{RB}{\pm}\mathsf{ATL}$. In this paper we provide the full proof that for single-agent, single-resource $\mathsf{RB}{\pm}\mathsf{ATL}^+$, the model-checking problem is Δ_2^{P}-complete. This result was originally stated in [40]. In the case of at least two agents, the model-checking problem is here proved to be in EXPTIME for a bounded number of resources. This result gives a tight complexity bound in the case of at least four resources, which is already known to be EXPTIME-hard [12, Cor. 1]. In all other cases there are no matching lower bounds, the problem is only known to be PSPACE-hard from ATL^+ [27].

2.2.3. Complexity of MC($RB{\pm}ATL^*$) *and fragments*

Unlike $\mathsf{RB}{\pm}\mathsf{ATL}$ and $\mathsf{RB}{\pm}\mathsf{ATL}^+$, tight complexity bounds are known for all variations of $\mathsf{RB}{\pm}\mathsf{ATL}^*$ on the number of agents and resources (see Figure 4). For at least two agents, the model-checking problem is 2EXPTIME-complete. The upper bound comes from [12, Th. 7], whereas the lower bound follows from the 2EXPTIME-hardness of ATL^*, which is proved by using two agents only [1]. On the other hand, the problem restricted to

a single agent becomes EXPSPACE-complete for an unbounded number of resources [12, Th. 8]; while for a bounded number r, model-checking $\mathsf{RB}{\pm}\mathsf{ATL}^*(\{1\}, r)$ is PSPACE-complete: the lower bound follows immediately from the PSPACE-hardness of the model-checking problem for CTL^*; as for the upper bound, we derive it from the fact that model-checking RBTL^* is in PSPACE [12, Cor. 2] and Theorem 19.

3. Decision Problems on 1-VASS

In this section, we present several decision problems for vector addition systems with states restricted to one counter (1-VASS) and we establish their PTIME upper bound. These results are most relevant to analyse the complexity of the model-checking problems for $\mathsf{RB}{\pm}\mathsf{ATL}$ and $\mathsf{RB}{\pm}\mathsf{ATL}^+$ restricted to a single resource and a single agent. Indeed, the restriction to a single agent entails that the strategy modalities state the existence of specific runs in 1-VASS. After recalling the main notions about VASS and 1-VASS, we present the nontermination and control-state reachability problems, and we introduce the new generalised control-state reachability problem. In particular, we prove that $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\}, 1))$ is Δ_2^{P}-complete (see Section 4.1) by using instances of this new generalised reachability problem on 1-VASS. We show that when restricted to a single counter, the new generalised control-state reachability problem can be solved in PTIME, thus generalising Theorems 3.3 and 3.5 from [39]. It is worth noting that the nontermination and control-state reachability problems are actually subproblems of the generalised problem. Nevertheless, their study and introduction have a two-fold motivation. Solving such problems is useful to solve the generalised problem and in particular to obtain the PTIME upper bound when restricted to 1-VASS. Moreover, in Section 5.2, we present a polynomial-time algorithm for $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}(\{1\}, 1))$ based on instances of the nontermination and control-state reachability problems, which we believe it is promising to obtain an efficient implementation, besides the fact that we are able to provide a self-contained version of the algorithm. Still, computational improvements are expected by taking advantage of the recent algorithmic results presented in [46].

3.1. Background on Vector Addition Systems with States

A *vector addition system with states (VASS)* [20] is a structure $V = \langle Q, r, R\rangle$, where Q is a finite set of *locations* (a.k.a., *control states*), $r \in \mathbb{N}$ is its *dimension* (also understood as its number of *counters*), R is a finite set of *transitions* in $Q \times \mathbb{Z}^r \times Q$. A *configuration* in a VASS V is a pair $\langle q, \vec{x}\rangle \in Q \times \mathbb{N}^r$. Note that the counter values are required to be non-negative. Given configurations $\langle q, \vec{x}\rangle$, $\langle q', \vec{x}'\rangle$ and a transition $t = q \xrightarrow{\vec{u}} q'$, we write $\langle q, \vec{x}\rangle \xrightarrow{t} \langle q', \vec{x}'\rangle$ whenever $\vec{x'} = \vec{u} + \vec{x}$. A *run* is defined as a sequence $\rho = \langle q_0, \vec{x}_0\rangle \xrightarrow{t_1} \langle q_1, \vec{x}_1\rangle \xrightarrow{t_2} \langle q_2, \vec{x}_2\rangle \cdots$ of configurations, where $\langle q_0, \vec{x}_0\rangle$ is the *initial* configuration. A run ρ is *simple* if no control state appears twice. A *finite path* is a sequence of contiguous transitions $t_1 \dots t_k$. It is worth stressing that paths do not take into account counter's values associated with transitions, while runs do. A *simple path* is a finite sequence of contiguous transitions $t_1 \dots t_k$ such that no control state occurs more than once. Hence, every finite run has an induced path but not every configuration $\langle q, \vec{x}\rangle$ can trigger a path leading to a run because the counter values have to be non-negative A *simple loop* is a sequence of contiguous transitions $t_1 \dots t_k$ such that the first control state of t_1 is equal to the second control state of t_k (and it occurs nowhere else) and no other control state

occurs more than once. An r-VASS is a VASS with $r \geq 1$ counters. By convention, a 0-VASS is a finite directed graph. In Figure 5, we present a graphical representation of a VASS with a single counter. The value on each edge corresponds to the update tuple/value.

VASS are known to be equivalent to Petri nets for many behavioural properties, which is particularly interesting, as Petri nets are ubiquitous computational models to handle concurrency. A famous result states that the reachability problem for Petri nets (and therefore for VASS) is decidable [47, 48, 49]. It has been the subject of the book [50] and its original proof requires many non-trivial steps involving graph theory, logic, and the theory of well-quasi-orderings. Nevertheless, the exact complexity of the reachability problem was open for decades: we know it is EXPSPACE-hard from [51], but non-elementarity has been shown only recently [52]. In this paper, we mainly deal with reachability questions for 1-VASS. So, the decision problems we consider are much more manageable than the general case with an arbitrary number of counters.

3.2. Nontermination and control-state reachability for 1-VASS

In order to show that the model-checking problem for the logic $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$ is PTIME-complete (the lower bound is inherited from PTIME-hardness of CTL model-checking, see e.g. [43, 24]), we establish that two well-known decision problems on vector addition systems with states (VASS), when restricted to a single counter, can be solved in polynomial time. We provide formal proofs below as it will be also useful to design algorithms, see Section 5.2. Besides, an energy problem equivalent to CREACH(1-VASS) introduced below is shown in PTIME in [32]. We begin by presenting two standard decision problems on VASS that play a crucial role in solving the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$ and $\mathsf{RB}{\pm}\mathsf{ATL}^{+}(\{1\},1)$.
Control-state reachability problem CREACH(VASS):

Input: a VASS V, a configuration $\langle q_0, \vec{x}_0\rangle$, and a control state q_f.

Question: is there a finite run with initial configuration $\langle q_0, \vec{x}_0\rangle$ and with final configuration with state q_f?

Nontermination problem NONTER(VASS):

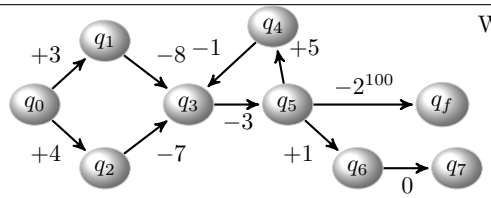
Input: a VASS V and a configuration $\langle q_0, \vec{x}_0\rangle$.

Question: is there an infinite run with initial configuration $\langle q_0, \vec{x}_0\rangle$?

In order to solve these problems, we need a few more definitions. In a 1-VASS, a simple loop is *positive* if the cumulated effect is positive. Given a run $\rho = \langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle, \ldots$ and $\alpha \geq 0$, we write $\rho^{+\alpha}$ to denote the sequence $\langle q_0, x_0 + \alpha\rangle, \ldots, \langle q_k, x_k + \alpha\rangle, \ldots$. If ρ is a run, the sequence $\rho^{+\alpha}$ is also a run (it is important to note that VASS have no zero-tests). The following lemma provides a characterisation of runs ending in a distinguished final state for 1-VASS.

Lemma 1. *Let V be an 1-VASS, $\langle q_0, x_0\rangle$ an initial configuration, and q_f a location. There is a finite run from $\langle q_0, x_0\rangle$ to the configuration $\langle q_f, x_f\rangle$, for some $x_f \geq 0$, iff (1) either $q_0 = q_f$, or*

2. there is a simple run $\langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ with $k > 0$ and $q_k = q_f$; or



Witness runs and path for $(V, \langle q_0, 7\rangle, q_f)$:

initial run: $\langle q_0, 7\rangle, \langle q_2, 11\rangle, \langle q_3, 4\rangle$

"> 0 loop": $\langle q_3, 4\rangle, \langle q_5, 1\rangle, \langle q_4, 6\rangle, \langle q_3, 5\rangle$

final path: $q_3 \xrightarrow{-3} q_5 \xrightarrow{-2^{100}} q_f$

Figure 5: Witness runs and path from Lemma 1(3)

3. *we have that*
   (a) *there is a simple run* $\langle q_0, x_0\rangle, \ldots, \langle q_n, x_n\rangle$,
   (b) *there is a positive simple loop* $t_1 \ldots t_\beta$ *such that* $\langle q_n, x_n\rangle \xrightarrow{t_1 \ldots t_\beta} \langle q_n, x_n + \alpha\rangle$ *is a run* ($\alpha > 0$),
   (c) *there is a simple path starting at* q_n *and ending at* q_f.

Note that in Lemma 1(3), without loss of generality, we can asssume that $n+\beta \leq |Q|$. As an illustration of Lemma 1(3), Figure 5 presents a 1-VASS V, and witness runs and path for the positive instance $(V, \langle q_0, 7\rangle, q_f)$ of CREACH(1-VASS). By contrast, the configuration $\langle q_0, 5\rangle$ cannot reach q_f.

Proof. First, it is not difficult to check that if either (1), (2) or (3) holds, then there is a finite run from $\langle q_0, x_0\rangle$ to a configuration $\langle q_f, x_f\rangle$, for some $x_f \geq 0$. By way of example, firing the strictly positive simple loop at least $|Q| \times \max\{|u| : q \xrightarrow{u} q' \text{ is a transition}\}$ times, allows us to pursue the run following the path from q_n to q_f.

Conversely, let us suppose that $\rho = \langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ is a run with $q_k = q_f$. If $q_0 = q_f$, then the witness run can be reduced to $\langle q_0, x_0\rangle$. Otherwise, either ρ is a simple run and condition (2) holds, or there are $0 \leq i < j \leq k$ such that $q_i = q_j$. In case $x_i \geq x_j$, the subrun $\langle q_i, x_i\rangle, \ldots, \langle q_j, x_j\rangle$ can be removed from ρ while leading to a run reaching q_f. Typically, the suffix subrun $\langle q_i, x_i\rangle, \ldots, \langle q_j, x_j\rangle, \ldots, \langle q_k, x_k\rangle$ with $\rho_\dagger = \langle q_j, x_j\rangle, \ldots, \langle q_k, x_k\rangle$ is replaced by $\rho_\dagger^{+\alpha}$ for $\alpha = x_i - x_j$. Such a transformation can be performed as soon as the subruns correspond to the application of simple loops with negative effect. Hence, without loss of generality, we can assume that ρ has no loop with negative effect.

If ρ is not a simple run, there are $0 \leq I < J \leq |Q|$ such that $q_I = q_J$ and $x_I < x_J$. We assume that I is minimal and we pick then a minimal J. Consequently,

(a) there is a simple run $\langle q_0, x_0\rangle, \ldots, \langle q_I, x_I\rangle$;

(b) there is a strictly positive simple loop $t_I \ldots t_{J-1}$ such that $\langle q_I, x_I\rangle \xrightarrow{t_I \ldots t_{J-1}} \langle q_I, x_I + (x_J - x_I)\rangle$;

(c) there is a path from q_J to $q_k = q_f$ such that $\langle q_J, x_J\rangle, \ldots, \langle q_k, x_k\rangle$ is a run. Indeed, suppose that $\langle q_J, x_J\rangle \xrightarrow{t_J \cdots t_{k-1}} \langle q_k, x_k\rangle$. Let $t'_1 \cdots t'_l$ be the path obtained from $t_J \cdots t_{k-1}$ by removing simple loops as much as possible. So, $t'_1 \cdots t'_l$ is a simple path from q_J to q_k.

As a result, condition (3) is satisfied and the lemma holds. □

The characterisation in Lemma 1 can be turned into an algorithm running in polynomial time that is used as a routine in the algorithm presented in Section 5.2.

Theorem 7. *The problem CREACH(1-VASS) is in* PTIME.

Proof. Let V be an 1-VASS, $\langle q_0, x_0\rangle$ an initial configuration, and q_f a location. If $q_0 = q_f$, we are done. Otherwise, define values $\mathtt{maxval}_q^i$ for $i \in [0, |Q|]$ and $q \in Q$ such that if there is a run $\langle q_0, x_0\rangle, \ldots, \langle q_j, x_j\rangle$ with $q_j = q$ and $j \leq i$, then the maximal value x_j among all these runs is precisely $\mathtt{maxval}_q^i$. When there is no such run, by convention $\mathtt{maxval}_q^i = -\infty$. Similar values have been considered to solve the boundedness problem for 1-VASS in [31]. Note also that [32, Section 3.3] shows that an energy problem equivalent to CREACH(1-VASS) can be solved in PTIME by using a similar approach, but with slightly different objects. Let us compute the values $\mathtt{maxval}_q^i$:

- $\mathtt{maxval}_{q_0}^0 \stackrel{\text{def}}{=} x_0$ and $\mathtt{maxval}_q^0 \stackrel{\text{def}}{=} -\infty$ for all $q \neq q_0$.
- For all q and $i+1 \in [1, |Q|]$,

$$\mathtt{maxval}_q^{i+1} \stackrel{\text{def}}{=} \max(\mathtt{maxval}_q^i, \{\mathtt{maxval}_{q'}^i + u \in \mathbb{N} \mid q' \xrightarrow{u} q \text{ is a transition}\}).$$

The values $\mathtt{maxval}_q^i$ can be computed in polynomial time in the size of V (the number $|Q|$ of locations being an essential parameter as well as the maximal absolute value $|u|$ from updates – integers being written in binary). One can show that $\mathtt{maxval}_q^i$ is indeed the maximal value as specified above. The proof is by induction on i. The base case for $i = 0$ is obvious and for the induction step, suppose that for all $j \leq i$ and $q \in Q$ $\mathtt{maxval}_q^j$ is the maximal counter value reaching q from $\langle q_0, x_0\rangle$ in at most j steps. Let us characterize the value $\mathtt{maxval}_q^{i+1}$, which is either the maximal value reached in at most i steps (i.e., equal to $\mathtt{maxval}_q^i$ by IH), or it is reached in exactly $i+1$ steps and is therefore a value of the form $\mathtt{maxval}_{q'}^i + u$ with some transition $q' \xrightarrow{u} q$. This is precisely the way $\mathtt{maxval}_q^{i+1}$ is computed above.

Given $q \in Q$, let $\mathcal{C}_q$ be the condition: there are a simple run $\langle q_0, x_0\rangle, \ldots, \langle q_n, x_n\rangle$ with $q_n = q$, and a positive simple loop $t_1 \ldots t_\beta$ such that $\langle q, x_n\rangle \xrightarrow{t_1 \ldots t_\beta} \langle q, x_n + \alpha\rangle$ is a run with $\alpha > 0$, $\beta > 0$ and $n + \beta \leq |Q|$. Note that (3) from Lemma 1 is equivalent to the existence of some $q \in Q$ such that $\mathcal{C}_q$ holds and there is a simple path from q to q_f. Moreover, observe that $\mathtt{maxval}_{q_f}^{|Q|} \neq -\infty$ implies (2) or (3) and that (2) implies $\mathtt{maxval}_{q_f}^{|Q|} \neq -\infty$.

One can show that $\mathcal{C}_q$ is equivalent to there being $I < J \leq |Q|$ such that:

(*a*) $\mathtt{maxval}_q^I \neq -\infty$.

(*b*) $\mathtt{maxval}_q^I < \mathtt{maxval}_q^J$ and $\mathtt{auxval}_q^0 < \mathtt{auxval}_q^{J-I}$, where the values $\mathtt{auxval}_{q'}^i$ ($i \in [0, J-I]$, $q' \in Q$) are defined as follows (similarly to what is done for $\mathtt{maxval}_{q'}^j$):

- $\mathtt{auxval}_q^0 \stackrel{\text{def}}{=} \mathtt{maxval}_q^I$ and $\mathtt{auxval}_{q'}^0 \stackrel{\text{def}}{=} -\infty$ for all $q' \neq q$.
- For all q' and $i+1 \in [1, J-I]$,

$$\mathtt{auxval}_{q'}^{i+1} \stackrel{\text{def}}{=} \max(\mathtt{auxval}_{q'}^i, \{\mathtt{auxval}_{q''}^i + u \in \mathbb{N} \mid q'' \xrightarrow{u} q' \text{ is a transition}\}).$$

Consequently, $\mathcal{C}_q$ can be checked in polynomial time in the size of V. Hence there is a finite run from $\langle q_0, x_0\rangle$ to the configuration $\langle q_f, x_f\rangle$, for some $x_f \geq 0$ iff either $q_f = q_0$ or $\texttt{maxval}_{q_f}^{|Q|} \neq -\infty$ or there is $q \in Q$ such that $\mathcal{C}_q$ holds and there is a simple path from q to q_f. Obviously, $\texttt{maxval}_{q_f}^{|Q|} \neq -\infty$ can be checked in PTIME, for each $q \in Q$, $\mathcal{C}_q$ can be checked in PTIME too with the above characterisation. Finally, existence of a path from q to q_f is an instance of the standard graph reachability problem GAP that is in NLOGSPACE. So, CREACH(1-VASS) is in PTIME. □

Remark. The values $\texttt{auxval}_{q'}^{i}$ in the proof of Theorem 7 are necessary to guarantee that the values $\texttt{maxval}_q^I$ and $\texttt{maxval}_q^J$ are obtained following a common subrun until reaching the configuration $\langle q, \texttt{maxval}_q^I\rangle$. In the proof of [31, Theorem 3.4] for solving the boundedness problem for 1-VASS in PTIME, a similar argument was needed but was missing. Here, we fix that result too with the help of the values $\texttt{auxval}_{q'}^{i}$. ■

Now, let us turn to the characterisation of runs and paths witnessing nontermination.

Lemma 2. *Let V be an 1-VASS and $\langle q_0, x_0\rangle$ an initial configuration. There is an infinite run starting at $\langle q_0, x_0\rangle$ iff*

1. *there is a simple run $\langle q_0, x_0\rangle, \ldots, \langle q_n, x_n\rangle$; and*
2. *there is a non-negative simple loop $t_1 \ldots t_k$ such that $\langle q_n, x_n\rangle \xrightarrow{t_1 \ldots t_k} \langle q_n, x_n + \alpha\rangle$ is a run ($\alpha \geq 0$).*

Proof. Clearly, the satisfaction of the two conditions implies that there is an infinite run starting at $\langle q_0, x_0\rangle$: just consider the run generated by $(t_1 \ldots t_k)^\omega$ from the configuration $\langle q_n, x_n\rangle$. Let us prove the other direction, similarly to what is done in the proof of Lemma 1. Suppose that $\rho = \langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle, \ldots$ is an infinite run. Without loss of generality, we can assume that ρ has no simple loop with negative effect. Unless (1) and (2) hold, there are $n \geq 0$ and $q \in Q$ such that $q_n = q$, $\{i \in \mathbb{N} \mid q_i = q\}$ is infinite and $\langle q_0, x_0\rangle, \ldots, \langle q_n, x_n\rangle$ is a simple run. Consider some $J > I$ such that $q_J = q_I = q$ (such indices I and J necessarily exist). Obviously, there is a non-negative simple loop $t_I \ldots t_{J-1}$ such that $\langle q_I, x_I\rangle \xrightarrow{t_I \ldots t_{J-1}} \langle q_J, x_J\rangle$ is a run. Hence, both conditions in the statement of the lemma hold. □

Once more, the characterisation in Lemma 2 can be turned into an algorithm to check nontermination, running in polynomial time. Note also that in Lemma 2, without loss of generality, we can asssume that $n + k \leq |Q|$.

Theorem 8. *The problem NONTER(1-VASS) is in* PTIME.

The proof has similarities with the proof of Theorem 7 (it may even sound a bit simpler), but there are a few subtle differences described below.

Proof. Let V be an 1-VASS and $\langle q_0, x_0\rangle$ an initial configuration. Define the values $\texttt{maxval}_q^i$ for $i \in [0, |Q|]$ and $q \in Q$ such that if there is a run $\langle q_0, x_0\rangle, \ldots, \langle q_i, x_i\rangle$ with $q_i = q$ (with i transitions), then the maximal value x_i among all these runs is precisely $\texttt{maxval}_q^i$. Note that these values are not the same as those from the proof of Theorem 7 as we consider runs of length *exactly* i. When there is no such run, by convention $\texttt{maxval}_q^i = -\infty$.

- $\texttt{maxval}^0_{q_0} \stackrel{\text{def}}{=} x_0$ and $\texttt{maxval}^0_q \stackrel{\text{def}}{=} -\infty$ for all $q \neq q_0$.
- For all q and $i+1 \in [1, |Q|]$,

  $$\texttt{maxval}^{i+1}_q \stackrel{\text{def}}{=} \max(\{\texttt{maxval}^i_{q'} + u \in \mathbb{N} \mid q' \xrightarrow{u} q \text{ is a transition, } \texttt{maxval}^i_{q'} \neq -\infty\}).$$

  By convention, the maximal value of the empty set is $-\infty$.

All the values $\texttt{maxval}^i_q$ can be computed in polynomial time in the size of V. One can show that $\texttt{maxval}^i_q$ is the maximal value as specified above. Finally, the characterisation in Lemma 2 is equivalent to: there are $q \in Q$ and $I < J \leq |Q|$ such that $\texttt{maxval}^I_q \neq -\infty$ and $\texttt{maxval}^I_q \leq \texttt{maxval}^J_q$ (this time, we do not require strictness, as in the proof of Theorem 7) and $\texttt{auxval}^0_q \leq \texttt{auxval}^{J-I}_q$, where the values $\texttt{auxval}^i_{q'}$ ($i \in [0, J-I]$, $q' \in Q$) are defined as

- $\texttt{auxval}^0_q \stackrel{\text{def}}{=} \texttt{maxval}^I_q$ and $\texttt{auxval}^0_{q'} \stackrel{\text{def}}{=} -\infty$ for all $q' \neq q$;
- for all q' and $i+1 \in [1, J-I]$,

  $$\texttt{auxval}^{i+1}_{q'} \stackrel{\text{def}}{=} \max\{\texttt{auxval}^j_{q''} + u \in \mathbb{N} \mid j \leq i,\ q'' \xrightarrow{u} q' \text{ is a transition}\}.$$

All conditions can be checked in polynomial time and therefore the nontermination problem for 1-VASS is in PTIME. □

More about computational complexity. Though we have shown that both problems restricted to 1-VASS are in PTIME, the control-state reachability problem for VASS is EXPSPACE-complete in general [51, 53]. Despite the fact that the control-state reachability problem is a subproblem of the covering problem that has been quite studied, including variants (see, e.g., [54, 55, 56]), [32, Section 3.3] handles the state-reachability problem for 1-VASS in PTIME and the uniform treatment we provided for control-state reachability and nontermination problems are the building blocks to design a tailor-made algorithm to solve the generalised control-state reachability problem restricted to 1-VASS. Above, we provided formal arguments for tractability by appropriately tuning and correcting the proof technique dedicated to the boundedness problem for 1-VASS from [31]. Note also that in [57], the updates in the *branching* VASS (BVASS) are restricted to the set $\{-1, 0, +1\}$ (see [57, Def. 1]). Therefore the upper bound in [57] does not extend to our present case where updates are arbitrary integers encoded in binary. When updates are arbitrary integers encoded in binary (as done herein), the relevant problems for 1-BVASS are known to be PSPACE-complete [45]. In the recent paper [46], new results about 1-VASS and extensions have been shown. For instance, the control-state reachability problem for 1-VASS is shown in NC (subclass of PTIME made of problems that can be solved in polylogarithmic parallel time), which is an improvement with respect to PTIME. Similarly, this problem for 1-VASS augmented with disequality tests is shown in PTIME [46]. The algorithm leading to the NC upper bound might be helpful to boost further the algorithm provided in Section 5.2.

3.3. Generalised control-state reachability for 1-VASS

Now, we present a new decision problem on VASS that plays a crucial role in deciding the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1)$. Before doing so, let us recall some helpful property about $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1))$: verifying $(M,s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi$ with a single agent in M, amounts to check the existence of an infinite computation λ starting from s, with initial budget $b \in \mathbb{N}$, satisfying Ψ. This latter requirement can be reformulated using the 1-VASS that can be constructed from M. Assuming that Ψ is a Boolean formula in negation normal form (NNF) built over atoms of the form $\mathsf{X}\, p$ or $p \,\mathsf{U}\, q$, the satisfaction of Ψ on λ amounts to guess which atomic formulae in Ψ hold, and then to check that the guess is correct. As $\neg(p \,\mathsf{U}\, q)$ is logically equivalent to $\mathsf{G}\, \neg q \vee (\neg q \,\mathsf{U}\, \neg p \wedge \neg q)$, $\neg \mathsf{X}\, p$ is logically equivalent to $\mathsf{X}\, \neg p$, and X and G distributes over $\wedge$, guessing which atomic formulae in Ψ hold amounts to guess a conjunction of the form

$$\mathsf{X}\, C_0 \wedge \mathsf{G}\, C'_0 \wedge (C_1 \,\mathsf{U}\, C'_1) \wedge \cdots \wedge (C_n \,\mathsf{U}\, C'_n)$$

for some $n \in \mathbb{N}$, where the C_i's and C'_j's are arbitrary conjunctions of literals. The conjunct $\mathsf{X}\, C_0$ can be checked at the first step, and handling $\mathsf{G}\, C'_0$ can be performed by restricting the model to states satisfying C'_0. The satisfaction of the atomic path formulae from the above conjunction (modulo the logical equivalences cited above) would make Ψ true propositionally. More precisely, let us define a natural satisfaction relation.

Definition 9. *Let Ψ be a positive Boolean combination of atomic path formulae of the form $C \,\mathsf{U}\, C'$, $\mathsf{X}\, C$ and $\mathsf{G}\, C$, where C and C' are conjunctions of literals and X be a finite set of atomic formulae of the form $C \,\mathsf{U}\, C'$, $\mathsf{X}\, C$ or $\mathsf{G}\, C$. We define the satisfaction relation $X \models \Psi$ based on the standard clauses below (which can be read as the conjunction of the atomic path formulae from X makes Ψ true propositionnally):*

$$\begin{array}{lll}
X \models C \,\mathsf{U}\, C' & \textit{iff} & C \,\mathsf{U}\, C' \in X \\
X \models \mathsf{X}\, C & \textit{iff} & \mathsf{X}\, C \in X \\
X \models \mathsf{G}\, C & \textit{iff} & \mathsf{G}\, C \in X \\
X \models \Psi'_1 \vee \Psi'_2 & \textit{iff} & X \models \Psi'_1 \textit{ or } X \models \Psi'_2 \\
X \models \Psi'_1 \wedge \Psi'_2 & \textit{iff} & X \models \Psi'_1 \textit{ and } X \models \Psi'_2
\end{array}$$

Indeed, $\bigwedge_{\Phi \in X} \Phi \Rightarrow \Psi$ is valid, whenever $X \models \Psi$. So, assuming that $X \models \Psi$, we can write $\bigwedge_{\Phi \in X} \Phi$ as a conjunction of the form

$$\mathsf{X}\, C_0 \wedge \mathsf{G}\, C'_0 \wedge (C_1 \,\mathsf{U}\, C'_1) \wedge \cdots \wedge (C_n \,\mathsf{U}\, C'_n)$$

since $(\mathsf{X}\, C \wedge \mathsf{X}\, C') \Leftrightarrow \mathsf{X}(C \wedge C')$ and $(\mathsf{G}\, C \wedge \mathsf{G}\, C') \Leftrightarrow \mathsf{G}(C \wedge C')$ are valid.

This clarification done, let us notice that the existence of an infinite computation λ starting from s, with initial budget $b \in \mathbb{N}$, satisfying the conjunction is equivalent to the existence of an infinite run with positions satisfying $C'_1, \ldots, C'_n$, while also imposing constraints on the satisfaction of the C_i's. There might be less than n distinct distinguished positions, in which case some witness position may satisfy two distinct C'_i and C'_j. Similarly, the ordering of positions satisfying respectively $C'_1, \ldots, C'_n$ may not coincide with the ordering $C'_1, \ldots, C'_n$. The definition of GREACH(VASS) below is designed based on these observations.

Generalised control-state reachability problem GREACH(VASS):

Input: a VASS V, a configuration $\langle q_0, \vec{x}_0\rangle$, a sequence $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ for some $\alpha \geq 0$ such that $X_1 \subseteq \cdots \subseteq X_\alpha \subseteq Q$ and $\{h_1, \ldots, h_\alpha\} \subseteq Q$. We also assume that $|\{q_0\} \cup \{h_1, \ldots, h_\alpha\}| = \alpha + 1$.

Question: is there an infinite run $\langle q_0, \vec{x}_0\rangle \to \langle q_1, \vec{x}_1\rangle \to \langle q_2, \vec{x}_2\rangle \cdots$ with indices $0 < i_1 < i_2 < \cdots < i_\alpha$ such that for all $j \in [1, \alpha]$, $q_{i_j} = h_j$ and $\{q_k \mid k < i_j\} \subseteq X_j$?

The condition $|\{q_0\} \cup \{h_1, \ldots, h_\alpha\}| = \alpha + 1$ could be removed leading also to a PTIME problem. However, it is included in our definition of GREACH(VASS) because, not only it simplifies technical developments, but more importantly, it is exactly what is needed in the proof of Theorem 11.

Here is an illustration of the witness run for $\alpha = 2$.

$$\overbrace{\underbrace{\langle q_0, \vec{x_0}\rangle \cdots \langle q_{i_1-1}, \vec{x_{i_1}-1}\rangle}_{\{q_0,\ldots,q_{i_1-1}\}\subseteq X_1}}^{\{q_0,\ldots,q_{i_1-1}\}\subseteq X_1} \to \underbrace{\langle q_{i_1}, \vec{x_{i_1}}\rangle \cdots \langle q_{i_2-1}, \vec{x_{i_2}-1}\rangle}_{\{q_0,\ldots,q_{i_2-1}\}\subseteq X_2 \text{ and } X_1\subseteq X_2} \to \langle q_{i_2}, \vec{x_{i_2}}\rangle \to \cdots$$

When $\alpha = 0$, the question is about the existence of an infinite run from $\langle q_0, \vec{x}_0\rangle$, which can be solved in PTIME for $r = 1$ by Theorem 8. Similarly, GREACH(VASS) restricted to 0-VASS (i.e., GREACH(0-VASS)) is NLOGSPACE-complete as the Graph Accessibility Problem (GAP) can be reduced to it and the NLOGSPACE upper bound can be established by showing that a positive instance requires the existence of a path of length at most $|Q| \times (\alpha + 2)$ whose constraints can be checked on-the-fly in NLOGSPACE.

Lemma 3. *GREACH(0-VASS) is* NLOGSPACE*-complete.*

Proof. An instance of GREACH(0-VASS) takes as input a finite graph $V = \langle Q, R\rangle$ with $R \subseteq Q \times Q$, $q_0 \in Q$, a sequence $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ for some $\alpha \geq 0$ such that $X_1 \subseteq \cdots \subseteq X_\alpha \subseteq Q$ and $\{h_1, \ldots, h_\alpha\} \subseteq Q$. The instance is positive iff there is an infinite path $q_0 \to q_1 \to q_2 \to \cdots$ with indices $0 < i_1 < i_2 < \cdots < i_\alpha$ such that for all $j \in [1, \alpha]$, $q_{i_j} = h_j$ and $\{q_k \mid k < i_j\} \subseteq X_j$.

The existence of such an infinite path is actually equivalent to the existence of a finite path $q_0 \to q_1 \to q_2 \to \cdots \to q_N$, with indices $0 = i_0 < i_1 < i_2 < \cdots < i_\alpha < N$ such that for all $j \in [1, \alpha]$, $q_{i_j} = h_j$ and $\{q_k \mid k < i_j\} \subseteq X_j$ with the following additional properties:

1. For $j \in [0, \alpha - 1]$, $q_{i_j} \to \cdots \to q_{i_{j+1}}$ is a simple path.
2. For some $\beta > i_\alpha$, $q_{i_\alpha} \to \cdots \to q_\beta$ is a simple path and $q_\beta \to \cdots \to q_N$ is a simple loop (whence, $q_N = q_\beta$).

Consequently, N can be bounded by $(\alpha+2)|Q|$. Solving GREACH(0-VASS) requires to guess a path of length at most $(\alpha+2)|Q|$ and one can check on-the-fly that it satisfies the relevant properties, whence the NLOGSPACE upper bound. NLOGSPACE-hardness is by a straightforward reduction from the Graph Accessibility Problem (GAP). □

Unlike 0-VASS, the addition of counters forbids a similar reduction of an instance of GREACH(VASS) to reachability questions on finite graphs.

Let V, $\langle q_0, \vec{x}_0\rangle$ and $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ be an instance of GREACH(1-VASS). When $\alpha > 0$, one can show that the instance is positive iff either (A) or (B) below holds.

Before presenting conditions (A) and (B), let us introduce some useful notation. Given an 1-VASS $V = \langle Q, r, R\rangle$ and $Q' \subseteq Q$, we write $V^{Q'}$ to denote the 1-VASS $\langle Q', r, R'\rangle$ such that $V^{Q'}$ is the restriction of V to Q', i.e., $R' \stackrel{\text{def}}{=} \{q \xrightarrow{u} q' \in R \mid q, q' \in Q'\}$. Here are conditions (A) and (B).

(A) The following conditions hold:

(a) For all $N \geq 0$, there is a run $\langle q_0, z_0\rangle \cdots \langle q_n, z_n\rangle$ with $\langle q_0, z_0\rangle = \langle q_0, x_0\rangle$, $q_n = h_1$, $z_n \geq N$, and $\{q_0, \ldots, q_{n-1}\} \subseteq X_1$.

(b) V', h_1 and $\langle X_2, h_2\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ is a positive instance of GREACH(0-VASS) where V' is the directed graph underlying V (integers on transitions are simply removed).

(c) There is a simple path from h_α to some location q' and there is a simple loop from q' with a non-negative cumulated effect.

(B) (a) above does not hold, and there is a simple run $\langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ in $V^{X_1 \cup \{h_1\}}$ with $q_k = h_1$ and V, $\langle h_1, x_{max}\rangle$ and $\langle X_2, h_2\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ is a positive instance of GREACH(1-VASS) with x_{max} the maximal counter value for reaching h_1 in $V^{X_1 \cup \{h_1\}}$ among all the simple runs from $\langle q_0, x_0\rangle$.

We briefly explain why the characterisation is correct (details follow) and leads to a PTIME upper bound. If (a) holds, one can reach the control state h_1 with a counter value as large as we want. The constraints in (b) on the subsequence $\langle X_2, h_2\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ induce reachability questions in the underlying directed graph V'. Similarly, the contraints in (c) express two simple reachability properties including the existence of a loop with non-negative cumulated effect. Otherwise, if (a) does not hold, the instance is positive only if there is a run $\rho = \langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ in $V^{X_1 \cup \{h_1\}}$ with $q_k = h_1$, and V, $\langle h_1, x_k\rangle$ and $\langle X_2, h_2\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ is a positive instance of GREACH(1-VASS). However, if not (a), we can assume that ρ is a simple run and the best we can do is to pick x_k equal to the maximal counter value for reaching h_1 among all the simple runs from $\langle q_0, x_0\rangle$ visiting only states in X_1. Checking (a), (b), and (c) can be done in PTIME (see Section 3.2). Similarly, computing x_{max} in (B) can be done in PTIME too.

Let us recapitulate about complexity. As GREACH(0-VASS)) is in NLOGSPACE, checking (A) can be done in PTIME. For checking (B), if (a) does not hold, then the computation of x_{max} can be done in polynomial time. A recursive call to an instance of GREACH(1-VASS) with a sequence strictly smaller allows us to get forthcoming Theorem 10.

Lemma 4. *Let V, $\langle q_0, x_0\rangle$ and $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ be an instance of GREACH(1-VASS) with $\alpha > 0$. It is a positive instance iff either (A) or (B) holds.*

Proof. Let V, $\langle q_0, x_0\rangle$ and $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ be an instance of GREACH(1-VASS) with $\alpha > 0$. We show the following properties.

(I) If (A) holds, then the instance is positive.

(II) If (B) holds, then the instance is positive.

(III) If the instance is positive, then either (A) or (B) holds.

(I) Assume that (A) holds. By the satisfaction of (b) and (c), there is a path in V' (i.e., the underlying finite graph of V), say $q'_0 \to q'_1 \to q'_2 \to \cdots \to q'_L$ such that $q'_0 = h_1$ and there are indices $0 < i_2 < \cdots < i_\alpha$ with for all $j \in [2, \alpha]$, $q'_{i_j} = h_j$ and $\{q'_k \mid k < i_j\} \subseteq X_j$. Moreover, for some $\beta > i_\alpha$, $q'_{i_\alpha} \to \cdots \to q'_\beta$ is a simple path, $q'_\beta \to \cdots \to q'_L$ is a simple loop, with non-negative effect if read from V. If $q'_0 \to q'_1 \to q'_2 \to \cdots \to q'_\beta$ corresponds to the path $t_1 \cdots t_\beta$ and $q'_\beta \to \cdots \to q'_L$ corresponds to the simple loop $t_{\beta+1} \cdots t_L$, the ω-sequence $t_1 \cdots t_\beta \cdot (t_{\beta+1} \cdots t_L)^\omega$ can lead (in V) to an infinite run from location h_2 whenever the initial counter value is greater than $||R|| \times (\alpha + 2)|Q|$, for $||R|| \stackrel{\text{def}}{=} \max\{|u| : q \xrightarrow{u} q' \in R\}$. Indeed, once the sequence of transitions $t_1 \cdots t_\beta \cdot t_{\beta+1} \cdots t_L$ can be fired from some initial configuration $\langle h_1, n\rangle$, the non-negative effect of $q'_\beta \to \cdots \to q'_L$ allows us to visit the loop infinitely. Having n at least equal to $||R|| \times (\alpha+2)|Q|$ allows us to be on the safe side (to maintain the counter value non-negative), as each transition in $t_1 \cdots t_\beta \cdot t_{\beta+1} \cdots t_L$ can decrement the counter by at most $||R||$ and $L \leq (\alpha+2)|Q|$. The satisfaction of (a) insures that location h_1 can be reached from $\langle q_0, x_0\rangle$ with a counter value greater than $||R|| \times (\alpha + 2)|Q|$ while preserving the constraints on X_1. The concatenation of the finite subrun leading to $\langle h_1, z\rangle$ (thanks to (a)) followed by the infinite run from $\langle h_1, z\rangle$ (thanks to (b),(c)) leads to an infinite run witnessing that the instance is positive.

(II) Assume that (B) holds. (a) does not hold and the control state h_1 can be reached with a maximal value x_{max} while visiting only states from X_1. Let $\langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ be the run in $V^{X_1 \cup \{h_1\}}$ with $q_k = h_1$ and x_k equal to x_{max}. As V, $\langle h_1, x_{max}\rangle$ and $\langle X_2, h_2\rangle, \ldots, \langle X_\alpha, h_\alpha\rangle$ is a positive instance of GREACH(1-VASS), the concatenation of $\langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ with the infinite witness run for that instance of GREACH(1-VASS) leads to a witness run for the instance V, $\langle q_0, x_0\rangle$ and $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$.

(III) Now suppose that V, $\langle q_0, x_0\rangle$ and $\langle X_1, h_1\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ is a positive instance of GREACH(1-VASS). Therefore, there is an infinite run $\langle q_0, x_0\rangle \to \langle q_1, x_1\rangle \to \langle q_2, x_2\rangle \cdots$ with indexes $0 < i_1 < i_2 < \cdots < i_\alpha$ such that for all $j \in [1, \alpha]$, $q_{i_j} = h_j$ and $\{q_k \mid k < i_j\} \subseteq X_j$.

We need to make a case analysis depending on whether (a) holds. Condition (a) states that h_1 can be reached from $\langle q_0, x_0\rangle$ with a counter value as great as desired, while visiting only states from X_1. If condition (a) does not hold, then there is a maximal counter value x_{max} and a run $\langle q'_0, y_0\rangle \cdots \langle q'_n, y_n\rangle$ in V with $\langle q'_0, y_0\rangle = \langle q_0, x_0\rangle$, $q'_n = h_1$, $y_n = x_{max}$, and $\{q'_0, \ldots, q'_{n-1}\} \subseteq X_1$. Necessarily, we have $x_{max} \geq x_{i_1}$. Let ρ be the run $\langle q_{i_1}, x_{i_1}\rangle \to \cdots \to \langle q_{i_2}, x_{i_2}\rangle \cdots$. Obviously, $\rho^{+(x_{max} - x_{i_1})}$ is also a run and actually it is a witness for the instance V, $\langle h_1, x_{max}\rangle$ and $\langle X_2, h_2\rangle, \ldots, \langle X_\alpha, h_\alpha\rangle$. Consequently, condition (B) holds true.

In the case condition (a) holds, $q_{i_1} \to q_{i_1+1} \to \cdots$ (the projection on Q of the suffix run $\langle q_{i_1}, x_{i_1}\rangle \to \langle q_{i_1+1}, x_{i_1+1}\rangle \to \cdots$) is a witness run for the instance V', h_1 and $\langle X_2, h_2\rangle$, $\ldots$, $\langle X_\alpha, h_\alpha\rangle$ of GREACH(0-VASS). Similarly, the suffix path $\rho' = q_{i_\alpha} \to q_{i_\alpha+1} \to \cdots$ is infinite ($h_\alpha = q_{i_\alpha}$), and therefore some state q' is repeated infinitely often. Consequently, there is a simple path from h_α to q' and a simple loop from q' to q'. It remains to show that the loop can be choosen to be non-negative so that condition (c) holds.

Suppose that the run ρ' contains only loops with negative effects. There is a location q_∞ that occurs infinitely often in the run, and therefore this leads to a contradiction as

$\mathbb{N}$ is well-founded (indeed any non-empty subrun starting by q_∞ and ending by q_∞ would decrease strictly the counter value). Consequently, ρ' contains a subrun $\rho'' = \langle q_L, x_L\rangle \to \langle q_{L+1}, x_{L+1}\rangle \to \cdots \langle q_{L'}, x_{L'}\rangle$ built over a non-negative loop and by removing from ρ' the subruns corresponding to a negative loop, we can assume that ρ'' is built from a non-negative simple loop (if ρ'' contains a subrun built over a simple non-negative loop, take it instead). So, we can assume w.l.o.g. that $\langle q_L, x_L\rangle \to \langle q_{L+1}, x_{L+1}\rangle \to \cdots \langle q_{L'}, x_{L'}\rangle$ is a simple non-negative loop. Consequently, there is a simple path from h_1 to $q_L = q_{L'}$ and there is a simple non-negative loop from q_L (the one obtained from ρ''), which entails (c). □

Theorem 10. *The problem GREACH(1-VASS) is in* PTIME.

Theorem 10 is one of the main contributions of the paper, as it is instrumental to determine the complexity of model checking $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\}, 1)$.

Proof. Let V, $\langle q_0, x_0\rangle$ and $\langle X_1, h_1\rangle$, …, $\langle X_\alpha, h_\alpha\rangle$ be an instance of GREACH(1-VASS) of size N. Here is a simple algorithm to solve GREACH(1-VASS).

- If condition (a) holds, then return 1 if (b),(c) hold, 0 otherwise.
- Otherwise ((a) does not hold),
  - if there is no simple run $\langle q_0, x_0\rangle, \ldots, \langle q_k, x_k\rangle$ in $V^{X_1 \cup \{h_1\}}$ with $q_k = h_1$ then return 0,
  - otherwise, compute x_{max} and return 1 if V, $\langle h_1, x_{max}\rangle$ and $\langle X_2, h_2\rangle$, …, $\langle X_\alpha, h_\alpha\rangle$ is a positive instance of GREACH(1-VASS) (0 otherwise). Note that we perform here a recursive call with a strictly shorter sequence $\langle X_2, h_2\rangle$, …, $\langle X_\alpha, h_\alpha\rangle$.

The following problems can be solved in polynomial time.

- As explained in the proof of Lemma 4, GREACH(0-VASS) and (c) can be checked in polynomial time, say in time $\mathcal{O}(p_1(N))$.
- Checking whether (a) holds can be done in polynomial time by using the data structures introduced in Section 3.2, say in time $\mathcal{O}(p_2(N))$. Indeed, (a) is equivalent to the satisfaction of the conditions below:
  - (a′) there is a simple run $\langle q_0, x_0\rangle, \ldots, \langle q_n, x_n\rangle$ in $V^{X_1 \cup \{h_1\}}$ and its projection over $X_1 \cup \{h_1\}$ is in the regular language $X_1^*(h_1 + \varepsilon)$;
  - (b′) there is a positive simple loop $t_1 \ldots t_\beta$ in $V^{X_1 \cup \{h_1\}}$ that can be fired from $\langle q_n, x_n\rangle$;
  - (c′) there is a simple path in $V^{X_1 \cup \{h_1\}}$ starting at q_n, ending at h_1;
  - (d′) if h_1 occurs in the simple loop in (b′), then $q_n = h_1$.

  Let us briefly show how (a′)–(d′) entail (a). Roughly speaking, the subrun is constructed as follows. From $\langle q_0, x_0\rangle$ consider the simple run leading to $\langle q_n, x_n\rangle$ (condition (a′)). Then, take the positive simple loop from (b′) $||R|| \times |Q|$ times reaching the configuration $\langle q_n, y\rangle$ with $y \geq ||R|| \times |Q|$. Then, consider the simple

run from $\langle q_n, y\rangle$ to $\langle h_1, z\rangle$ based on the transitions of the simple path from the condition (c′). As this is a simple run, $z \geq 0$ while preserving the constraints on X_1. For the proof in the other direction, this boils down to detect a positive simple loop. The details are omitted here.

- Similarly, when the condition (a) does not hold, computing x_{max} (if it exists) can also be computed in polynomial time using the proofs in Section 3.2, say in time $\mathcal{O}(p_3(N))$.

From the above algorithm, checking whether (a) holds is done at most α times, as well as computing x_{max}. Consequently, the algorithm runs in time $\mathcal{O}(\alpha(p_2(N^2) + p_3(N^2)) + p_1(N^2))$. The value N^2 is due to the fact that every time a new value x_{max} is computed, it may add the initial size N (in the worst case) to the size of the new input and this can be done at most α times with $\alpha \leq N$. □

The PTIME upper bound in Theorem 10 is a drastic drop compared to the complexity for the general problem GREACH(VASS). Indeed, GREACH(VASS) is EXPSPACE-hard as the control-state reachability problem for VASS can be reduced to it (and then we use [51]). GREACH(VASS) is definitely in EXPSPACE by [58, Theorem 5.4] that deals with the linear-time μ-calculus on VASS. Besides, for all $r \geq 1$, model-checking r-VASS with linear-time μ-calculus has been shown in PSPACE [58, Theorem 4.1] (PSPACE-hardness still holds with a unique counter, inherited from plain LTL model-checking [59]). Herein, we established that GREACH(1-VASS) behaves even better, as it can be solved in polynomial time only. Theorem 10 shall be essential for the results shown in Section 4 below.

4. The Model-Checking Problem for RB±ATL$^+$

In this section, we mainly consider the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\}, 1)$ as well as for $\mathsf{RB}{\pm}\mathsf{ATL}^+(r)$, for some fixed $r \geq 1$.

To this end, we need to provide a few more introductory notions. Given an RB-CGS $M = \langle\{1\}, 1, S, Act, act, wf, \delta, L\rangle$ with a single agent and a single resource, we define the 1-VASS $V_M = \langle S, 1, R_V\rangle$ such that $q \xrightarrow{u} q' \in R_V$ iff there is some action $\mathtt{a} \in act(q, 1)$ such that $\delta(q, \mathtt{a}) = q'$ and $wf(q, 1, \mathtt{a}) = u$. Similarly, we write $K_M = \langle S, R, L\rangle$ to denote the Kripke-style structure such that $q\ R\ q'$ iff there is some action $\mathtt{a} \in act(q, 1)$ such that $\delta(q, \mathtt{a}) = q'$. Observe that M and V_M share the same labelling function L. We introduce Kripke-style structures as the modality $\langle\!\langle\{1\}^\omega\rangle\!\rangle$ amounts to forgetting about weights in M, and therefore M can be understood as the Kripke-style structure K_M, and $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\}, 1)$ model-checking reduces to CTL^+ model-checking. Similarly, the strategy modality $\langle\!\langle\emptyset^b\rangle\!\rangle$ behaves as the universal path quantifier A in weight-free transition systems, and therefore the model-checking problem reduces again to CTL^+ model-checking. Given $S_1 \subseteq S$, we write $V_M^{S_1}$ (resp. $K_M^{S_1}$) to denote the restriction of V_M (resp. K_M) to the locations/states in S_1 only.

4.1. Model-checking problem for RB±ATL$^+$($\{1\}, 1$): algorithm

To solve MC($\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\}, 1)$), there is an essential case to consider, namely how to verify whether $(M, s) \models \langle\!\langle\{1\}^b\rangle\!\rangle\Psi$ (this is the key case that needs to be solved in a satisfactory way, complexity-wise). With a single agent in M, this amounts to checking the existence of a computation λ starting from s, with initial budget $b \in \mathbb{N}$, satisfying Ψ.

This is the place where the problem GREACH(1-VASS) introduced in Section 3.3 helps. The conditions on Ψ in Lemma 5 below are relaxed at a later stage.

Lemma 5. *Let Ψ be a Boolean formula built over path formulae of the form $\mathsf{X}\,p$ or $p\,\mathsf{U}\,q$, where p, q are propositional variables. Let M be an RB-CGS, $s \in S$ and $b \in \mathbb{N}$. The problem of checking $(M, s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi$ (in $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\}, 1)$) is in* NP.

Proof. We recall that verifying $(M, s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi$ with a single agent in M amounts to check the existence of an infinite computation λ starting from s, with initial budget $b \in \mathbb{N}$, satisfying Ψ. Without loss of generality, one can assume that Ψ is a Boolean formula in negation normal form (NNF) built over atomic formulae of the form $\mathsf{X}\,p$ or $p\,\mathsf{U}\,q$. Satisfaction of Ψ on λ amounts to guess which atomic formulae in Ψ hold, and then to check that the guess is correct. The verification step is done in PTIME, so that the whole step of model-checking $\langle\!\langle \{1\}^b \rangle\!\rangle \Psi$ can be done in NP.

Since $\neg(p\,\mathsf{U}\,q)$ is logically equivalent to $\mathsf{G}\neg q \vee (\neg q\,\mathsf{U}(\neg p \wedge \neg q))$, $\neg\mathsf{X}\,p$ is logically equivalent to $\mathsf{X}\neg p$, and X and G distributes over $\wedge$, the formula Ψ can be assumed to be a positive Boolean formula built over atomic formulae of the form $\mathsf{X}\,C$, $\mathsf{G}\,C'$ or $C\,\mathsf{U}\,C'$ where C and C' are conjunctions of literals. Guessing which atomic formulae in Ψ hold to evaluate Ψ to true, amounts to guess a conjunction of the form

$$\mathsf{X}\,C_0 \wedge \mathsf{G}\,C'_0 \wedge (C_1\,\mathsf{U}\,C'_1) \wedge \cdots \wedge (C_n\,\mathsf{U}\,C'_n), \tag{5}$$

for some $n \in \mathbb{N}$, where the C_i's and C'_i's are conjunctions of literals (empty conjunctions are equivalent to $\top$). Below, we treat only the cases with C_0 and C'_0 distinct from $\top$ and $n > 0$, but the other cases admit a simpler treatment (omitted herein). Of course, the satisfaction of the atomic path formulae from the above conjunction (modulo the logical equivalences cited above) would make Ψ true propositionally (see Section 3.3 for a detailed description of the conditions whereby the conjunction enforces the truth of Ψ, see also Definition 9). Existence of an infinite computation λ starting from s, with initial budget $b \in \mathbb{N}$, satisfying conjunction (5) is equivalent to the existence of an infinite run $\langle q_0, x_0\rangle \to \langle q_1, x_1\rangle \to \langle q_2, x_2\rangle \cdots$ in the 1-VASS V_M satisfying the following conditions:

1. $\langle q_0, x_0\rangle = \langle s, b\rangle$.
2. $\{q_i \mid i \in \mathbb{N}\} \subseteq \{s \in S \mid (M, s) \models C'_0\} \stackrel{\text{def}}{=} S_1$. Hence, we can consider $V_M^{S_1}$ instead of V_M.
3. $(M, q_1) \models C_0$. As there is only a linear amount of configurations obtained from $\langle q_0, x_0\rangle$ in one step, we can easily get rid of the constraint on $\langle q_1, x_1\rangle$ by exploring all possibilities (and this does not cause any exponential blow-up). Let $I_0 = \{i \in [1, n] \mid (M, q_0) \models C'_i\}$ and $I_1 = \{i \in [1, n] \mid (M, q_1) \models C'_i \text{ and } (M, q_0) \models C_i\}$.
4. There are a partition $\{Y_1, \ldots, Y_\alpha\}$ of $[1, n] \setminus (I_0 \cup I_1)$ and a strict linear ordering $0 < i_1 < \ldots < i_\alpha$ such that for all $j \in [1, \alpha]$, we have $(M, q_{i_j}) \models \bigwedge_{k \in Y_j} C'_k$ and $\{q_\beta \mid \beta < i_j\} \subseteq \{s \mid (M, s) \models \bigwedge_{k \in Y_j} C_k\}$.

   In a nutshell, the satisfaction of $(C_1\,\mathsf{U}\,C'_1) \wedge \cdots \wedge (C_n\,\mathsf{U}\,C'_n)$ requires n witness positions, non necessarily pairwise different, whence the partition $\{Y_1, \ldots, Y_\alpha\}$. The satisfaction of the C_k's (first arguments of the U-formulae) is constrained by the implicit ordering $Y_1, \ldots, Y_\alpha$ and by the positions $i_1, \ldots, i_\alpha$. This is a standard type of reasoning when CTL^+ is involved, see e.g., [24].

Consequently, after guessing which atomic formulae hold in the path formula Ψ, the next configuration $\langle q_1, x_1\rangle$, the partition $Y_1, \ldots, Y_\alpha$ (with linearly ordered sets), the sequence of states $q_{i_1}, \ldots, q_{i_\alpha}$, we have then to solve an instance of GREACH(1-VASS) for the 1-VASS $V_M^{S_1}$ and initial configuration $\langle q_1, x_1\rangle$. We write X_j to denote the set $\{s \mid (M,s) \models \bigwedge_{k\in Y_j} C_k\}$. Without any loss of generality, we can assume that q_1, $q_{i_1}, \ldots, q_{i_\alpha}$ are all pairwise distinct. For instance, if $q_{i_k} = q_{i_l}$, then we can simply merge Y_k with Y_l. So we have to solve an instance of GREACH(1-VASS) of the form $V_M^{S_1}$, $\langle q_1, x_1\rangle$ and $\langle X_1, q_{i_1}\rangle, \ldots, \langle X_\alpha, q_{i_\alpha}\rangle$, which can be done in PTIME by Theorem 10. The guess and check steps can be therefore performed in NP as all the witnesses are of polynomial size in the size of the inputs, and all the checking steps can be done in polynomial time in full generality. □

Hence, the whole model-checking algorithm for $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1)$ can be shown in Δ_2^{P} as it uses a polynomial amount of instances of the problem in Lemma 5. Each instance can be solved in NP, since GREACH(1-VASS) is in PTIME (Theorem 10).

Theorem 11. MC($\mathit{RB}{\pm}\mathit{ATL}^+(\{1\},1)$) *is* Δ_2^{P}*-complete.*

Proof. As regards the lower bound, it follows from the Δ_2^{P}-hardness of model-checking CTL^+ [30]. As for the upper bound, let $M = \langle\{1\}, 1, S, \mathit{Act}, \mathit{act}, \mathit{wf}, \delta, L\rangle$ be an RB-CGS, and ϕ a formula in $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1)$. Let us present Algorithm 1 that computes the finite set $\{s \in S \mid (M,s) \models \phi\}$, where we assume that $b \in \mathbb{N}$ and Ψ is a Boolean formula in NNF built over path formulae of the form $\mathsf{X}\,\phi$ or $\phi\,\mathsf{U}\,\phi'$, for state formulae ϕ, ϕ'. Moreover, let us assume that the maximal state formulae occurring in Ψ are $\phi_1, \ldots, \phi_N$ and the model-checking algorithm has already determined that they hold true exactly on the states in sets $S_1^\star, \ldots, S_N^\star$, respectively. Let $\Psi^\star$ be the formula obtained from Ψ by replacing each ϕ_i by a fresh propositional variable p_i, and $M^\star$ be the RB-CGS obtained from M by modifying the labelling function as follows: $L^\star(p_i) \stackrel{\text{def}}{=} S_i^\star$. We have the following equivalences:

1. $(M,s) \models \langle\!\langle \emptyset^b \rangle\!\rangle \Psi$ iff $(K_{M^\star}, s) \models \mathsf{A}\Psi^\star$ in CTL^+,
2. $(M,s) \models \langle\!\langle \{1\}^\omega \rangle\!\rangle \Psi$ iff $(K_{M^\star}, s) \models \mathsf{E}\Psi^\star$ in CTL^+.
3. Concerning the case $(M,s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi$ for $b \in \mathbb{N}$, we have $(M,s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi$ iff $(M^\star, s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi^\star$, which can be checked in NP by Lemma 5.

By structural induction, one can show that $(M,s) \models \phi$ iff $s \in \mathrm{MC}(M,\phi)$.

As far as computational complexity is concerned, $\mathrm{MC}(M,\phi)$ is computed with a recursion depth linear in the size of ϕ and a polynomial number of NP calls. More precisely, for each occurrence of a subformula ψ of ϕ, $\mathrm{MC}(M,\psi)$ can be computed only once, which guarantees the overall number of calls of the form $\mathrm{MC}(M,\psi)$: it is sufficient to take advantage of dynamic programming and to work with a table to remember the values $\mathrm{MC}(M,\psi)$ already computed. For the sake of clarity such a mechanism is omitted in the present algorithm. Intuitively, we would handle an array T, where $T[\psi]$ takes either the value $\perp$ (undefined) or a subset of S. Initially, all the values of T are undefined. Whenever $\mathrm{MC}(M,\psi)$ is invoked in the algorithm above, we operate a change of the following form in the code: we first check whether $T[\psi]$ is defined. It is only in the case $T[\psi]$ is undefined that a recursive call $\mathrm{MC}(M,\psi)$ is performed. This technique is standard and herein we use it so that for each subformula ψ, $\mathrm{MC}(M,\psi)$ is actually

Algorithm 1 – $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1)$ Model-checking –

procedure $\mathrm{MC}(M,\phi)$
  case ϕ **of**
  p: **return** $\{s \in S \mid s \in L(p)\}$
  $\neg\psi$: **return** $S \setminus \mathrm{MC}(M,\psi)$
  $\phi_1 \wedge \phi_2$: **return** $\mathrm{MC}(M,\phi_1) \cap \mathrm{MC}(M,\phi_2)$
  $\langle\!\langle \emptyset^b \rangle\!\rangle \Psi / \langle\!\langle \emptyset^\omega \rangle\!\rangle \Psi$: **return** $\{s \mid (K_{M^\star}, s) \models \mathsf{A}\Psi^\star \text{ in } \mathsf{CTL}^+\}$
  $\langle\!\langle \{1\}^\omega \rangle\!\rangle \Psi$: **return** $\{s \mid (K_{M^\star}, s) \models \mathsf{E}\Psi^\star \text{ in } \mathsf{CTL}^+\}$
  $\langle\!\langle \{1\}^b \rangle\!\rangle \Psi$: **return** $\{s \mid (M^\star, s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \Psi^\star$ with the algorithm in the proof of Lemma 5 $\}$
  end case
end procedure

called only once. We recall that Δ_2^{P}, a.k.a., $\textsc{ptime}^{\textsc{np}}$, is precisely the class of problems that can be solved in polynomial time with an oracle in NP. We also take advantage of the fact that the model-checking problem for CTL^+ is in Δ_2^{P} (see, e.g., [24]). □

The decision procedure for solving the model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1)$ invokes the subroutine for solving the model-checking problem for CTL^+ with one counter, that be run optimally thanks to Lemma 5. The existence of a polynomial-time reduction between $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1))$ and the model-checking problem for CTL^+ is guaranteed, as both problems are Δ_2^{P}-complete. The question about the encoding of $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1))$ into plain CTL^+ is certainly interesting and solving conditions (A) and (B) in Lemma 4 actually amounts to detect graph-theoretical properties on the 1-VASS, so a direct encoding is feasible, but it is not further investigated here.

4.2. Model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+(r)$

In [12], the 2EXPTIME upper bound for model-checking $\mathsf{RB}{\pm}\mathsf{ATL}$ is obtained with a decision procedure calling subroutines to solve the nontermination and the control-state reachability problems for AVASS on instances of the form $\mathcal{A}_{M,A,s^\star}$, where M is an RB-CGS, A is a coalition, and $s^\star$ is a state in M, as described in Section 4.2.1 below. On the other hand, to obtain the 2EXPTIME upper bound for model-checking $\mathsf{RB}{\pm}\mathsf{ATL}^*$, also in [12], the decision procedure calls a subroutine for solving the parity game problem for AVASS (see Section 4.2.1) but, in the worst-case, on systems with a doubly-exponential number of locations in the size of the input formula. Indeed, synchronised products are considered between deterministic parity automata (on ω-words) and AVASS of the form $\mathcal{A}_{M,A,s^\star}$. This is due to the fact that given an LTL formula, an equivalent deterministic parity automaton might have a doubly-exponential number of states in the size of the input LTL formula (see, e.g., [60, 61]). Hence, even when the number of resources r is fixed (which is an assumption made in this section), solving $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(r))$ by using the method in [12], would lead to a 2EXPTIME upper bound.

In this section, we explain how to gain one exponential by providing a decision procedure that solves $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(r))$ in exponential time, when r is fixed (this is known to be optimal as soon as $r \geq 4$). Such an improvement can be explained by the fact that reasoning about LTL formulae of temporal depth one is usually simpler than for arbitrary LTL formulae (see, e.g., [62, Section 7.3]). More precisely, we are able to de-

sign some ad-hoc verification method using alternating VASS without going through a translation from LTL formulae to deterministic parity automata. Apart from our new EXPTIME bound, this section can be viewed as providing an alternative decision procedure for solving MC(RB±ATL$^+$) without going through the determinisation of Büchi automata, or as generalising the reduction used to decide RB±ATL [12], but at the cost of building AVASS with an exponential number of locations (which is strictly more expensive than for RB±ATL, but strictly less than for RB±ATL*). As CTL$^+$ is exponentially more succinct than CTL [63], which entails that some ATL$^+$ (resp. RB±ATL$^+(Ag, r)$) formulae can be exponentially more succinct than ATL (resp. RB±ATL(Ag, r)) formulae, this extra cost for solving MC(RB±ATL$^+(r)$) is most likely the best we can hope for.

4.2.1. Correspondence between RB-CGS and AVASS

In this section, we recall the notion of alternating vector addition systems with states (AVASS), as well as relevant related decision problems. Roughly speaking, AVASS can be viewed as VASS in which fork rules are also allowed, but such branching in computations has no effect on the counters. As shown in [12], AVASS are operational models closely related to the model-checking problem for RB±ATL-like logics. First, we need to introduce some preliminaries.

A *binary tree* T, which may contain nodes with a single child, is a non-empty subset of $\{1, 2\}^*$ such that, for all $\mathfrak{n} \in \{1, 2\}^*$ and $i \in \{1, 2\}$, $\mathfrak{n} \cdot i \in T$ implies $\mathfrak{n} \in T$ and, $\mathfrak{n} \cdot 2 \in T$ implies $\mathfrak{n} \cdot 1 \in T$. Trees of arbitrary finite arity are defined accordingly (details are omitted herein).

Definition 12 (AVASS [21]). *An* alternating VASS *is a tuple* $\mathcal{A} = \langle Q, r, R_1, R_2\rangle$ *such that* Q *is a finite set of* locations*;* $r \geq 0$ *is the number of resources;* R_1 *is a finite subset of* $Q \times \mathbb{Z}^r \times Q$ *(*update rules*); and* R_2 *is a (finite) subset of* $Q \times Q \times Q$ *(*fork rules*).*

Standard VASS [20] are AVASS with no fork rules (i.e., $R_2 = \emptyset$). A *derivation skeleton* in $\mathcal{A}$ is a labelling $\mathcal{D} : T \to (R_1 \cup R_2 \cup \{\bot\})$ such that T is a binary tree, and if $\mathfrak{n}$ has one child (resp. has two children, is a leaf) in T, then $\mathcal{D}(\mathfrak{n}) \in R_1$ (resp. $\mathcal{D}(\mathfrak{n}) \in R_2$, $\mathcal{D}(\mathfrak{n}) = \bot$). A *derivation* in $\mathcal{A}$ based on $\mathcal{D}$ is a labelling $\hat{\mathcal{D}} : T \to Q \times \mathbb{Z}^r$ such that:

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

So, the fork rules do not update the value of resources. Hence, there is an asymmetry between update rules and fork rules. Differently from branching VASS (see, e.g., [64, 65, 57, 45, 66]), fork rules have no effect on counter values. A derivation $\hat{\mathcal{D}}$ is *admissible* (or a *proof*) whenever only natural numbers occur in it. By way of example, a proof can be found below (with the root at the bottom) assuming that $R_1 = \{q_1 \xrightarrow{\langle -2,3\rangle} q_0, q_2 \xrightarrow{\langle 4,4\rangle} q_3\}$

and $R_2 = \{q_0 \to q_1, q_2\}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\langle q_1, \langle 3,8\rangle\rangle} \quad \langle q_2, \langle 3,8\rangle\rangle}{\langle q_0, \langle 3,8\rangle\rangle}}{\langle q_1, \langle 5,5\rangle\rangle} \quad \dfrac{\langle q_3, \langle 9,9\rangle\rangle}{\langle q_2, \langle 5,5\rangle\rangle}}{\dfrac{\langle q_0, \langle 5,5\rangle\rangle}{\langle q_1, \langle 7,2\rangle\rangle}}$$

Hereafter, we consider the following decision problems pertaining to AVASS.

Control-state reachability: given an AVASS $\mathcal{A}$ and control states q_0 and q_f, is there a finite proof with root labelled by $\langle q_0, \vec{0}\rangle$ and each leaf belonging to $\{q_f\} \times \mathbb{N}^r$?

Nontermination: given an AVASS $\mathcal{A}$ and a control state q_0, is there a proof whose root is labelled by $\langle q_0, \vec{0}\rangle$ and all the maximal branches are infinite?

Parity game problem: given an AVASS $\mathcal{A}$, a state $q_0 \in Q$, $\vec{b} \in \mathbb{N}^r$, and a colouring $\mathtt{col} : Q \to [0, \mathfrak{p} - 1]$ for some number $\mathfrak{p} \geq 1$, is there a proof with root labelled by $\langle q_0, \vec{b}\rangle$, all the maximal branches being infinite, and the maximal colour occurring infinitely often along each maximal branch being even?

These problems are known to be 2EXPTIME-complete [21, 22, 25]. Decidability of control-state reachability and nontermination problems were first established in [67] by using monotonicity of the games. The 2EXPTIME upper bound is preserved if we assume that the root is labelled by $\langle q_0, \vec{b}\rangle$ with $\vec{b} \in \mathbb{N}^r$ encoded in binary or if the set of fork rules R_2 is a finite subset of $\bigcup_{\beta \geq 2} Q^\beta$ (by suitably adapting all the definitions, see e.g., [12, Lemma 7]). Elements in Q^β are said to be $(\beta - 1)$-ary ($\beta - 1$ is the number of branches in the forking).

By [25, Corollary 5.7] on the parity energy game problem with initial credit, and by [23, Lemma 4] relating the parity game problem for AVASS and the parity energy game problem, the former can be solved in time

$$(|Q| \times ||R_1||)^{2^{\mathcal{O}(r \times log(r+\mathfrak{p}))}} + \mathcal{O}(r \times log||\vec{b}||),$$

where $||\vec{b}|| \stackrel{\text{def}}{=} \max\{|\vec{b}[i]| : i \in [1, r]\}$ and $||R_1|| \stackrel{\text{def}}{=} \max\{||\vec{u}|| : q \xrightarrow{\vec{u}} q' \in R_1\}$. When r and $\mathfrak{p}$ are bounded, the problem is therefore in EXPTIME. Indeed, the expression $2^{\mathcal{O}(r \times log(r+\mathfrak{p}))}$ and r are bounded, whereas $||R_1||$ is at most exponential in the size of the input (integers are encoded in binary).

Now, we briefly recall the correspondence established in [12] between strategies in RB±ATL-like logics and proofs in AVASS. Since we use this correspondence in the paper, we explain it in detail for the reader's benefit. Below, we consider AVASS with fork rules in $\bigcup_{\beta \geq 2} Q^\beta$ (arbitrary arity), and where proofs are trees with nodes labelled by elements in $Q \times \mathbb{N}^r$. Let M be a finite RB-CGS, $A \subseteq Ag$ be a coalition, and $s^\star$ be one of its states. We construct the AVASS $\mathcal{A}_{M,A,s^\star}$ such that the set of computations in M starting in $s^\star$ and respecting a strategy F_A corresponds to a derivation skeleton whose root is labelled by an update rule with first state $s^\star$. Hence, we leverage on the fact that a $\vec{b}$-strategy

generates a set of maximal computations that can be arranged as a finitely-branching tree with infinite branches only, and such an infinite tree can be viewed precisely as an infinite proof on $\mathcal{A}_{M,A,s^\star}$.

Given $M = \langle Ag, r, S, Act, act, wf, \delta, L\rangle$ and a distinguished state $s^\star \in S$, the AVASS $\mathcal{A}_{M,A,s^\star} \stackrel{\text{def}}{=} \langle Q, r, R_1, R_2\rangle$ is built as follows:

$$\begin{array}{rcl} Q & \stackrel{\text{def}}{=} & \{s^\star\} \cup \{\langle s, f\rangle \mid s \in S,\ f \in D_A(s)\} \cup \\ & & \{\langle g, s'\rangle \mid s', s'' \in S,\ g \in D_{Ag}(s''),\ \delta(s'', g) = s'\}. \end{array}$$

Every location in Q is attached to a state in S and contains a finite piece of information: in $\langle s, f\rangle$, f is a joint action available to A implementing its strategy whereas in $\langle g, s'\rangle$, we remember the global joint action to reach s'.

- The set R_1 of update rules contains the following elements.
  - $\langle s^\star, wf_A(s^\star, f), \langle s^\star, f\rangle\rangle$, for all $f \in D_A(s^\star)$. Indeed, the locations in Q attached to $s^\star$ have a special treatment.
  - $\langle\langle g, s\rangle, wf_A(s, f), \langle s, f\rangle\rangle$, for all $\langle g, s\rangle \in Q$ and $f \in D_A(s)$.
- The set R_2 of fork rules contains the following elements.
  - For $\langle s, f\rangle \in Q$, let $\{\langle g_1, s_1\rangle, \ldots, \langle g_\alpha, s_\alpha\rangle\} = \{\langle g, s'\rangle \in S \mid s' = \delta(s, g),\ g \in D_{Ag}(s),\ f \sqsubseteq g\}$. This set is non-empty because the protocol function always returns a non-empty set of actions. We recall that a protocol function in an RB-CGS is a map of the form $act : S \times Ag \to (\wp(Act) \setminus \emptyset)$.

    We add the α-ary fork rule $\langle\langle s', f\rangle, \langle g_1, s_1\rangle, \ldots, \langle g_\alpha, s_\alpha\rangle\rangle$ to R_2. The joint actions available for the opponent coalition $(Ag \setminus A)$ determine which locations $\langle g_1, s_1\rangle$, $\ldots$, $\langle g_\alpha, s_\alpha\rangle$ can be reached. It is the proponent restriction condition that guarantees that using fork rules in this place (i.e., without updating the counter values) is correct. We recall that the proponent restriction condition enforces that the resource availability depends only on the agents in proponent coalitions.

Given an infinite computation $\lambda = s_0 \stackrel{g_1}{\to} s_1 \stackrel{g_2}{\to} s_2 \ldots$ starting in $s^\star = s_0$, also respecting a strategy F_A, we can associate it with an infinite sequence (which we call an *extended computation*)

$$\mathtt{ext}(\lambda, A) \stackrel{\text{def}}{=} s_0 \stackrel{\vec{u}_0}{\to} \langle s_0, f_0\rangle \to \langle g_0, s_1\rangle \stackrel{\vec{u}_1}{\to} \langle s_1, f_1\rangle \to \ \cdots$$

with $s_0 = s^\star$, and for all $n \geq 0$, $f_n = F_A(s_0 \stackrel{g_1}{\to} s_1 \ldots \stackrel{g_n}{\to} s_n)$ where f_n is the restriction of g_n to A and $wf_A(s_n, f_n) = \vec{u}_n$. Hence, the way $\mathtt{ext}(\lambda, A)$ is designed from λ essentially depends on the weight of the actions fired by the agents in the coalition A. Typically, $A \neq A'$ may lead to $\mathtt{ext}(\lambda, A)$ different from $\mathtt{ext}(\lambda, A')$ (but not necessarily).

Further, transitions in M can be viewed as triples $\langle s', g, s''\rangle$ such that $\delta(s', g) = s''$, also written as $s' \stackrel{g}{\to} s''$. The finite set of such transitions is denoted by Σ_M, which can be interpreted as a finite alphabet when M is finite. An infinite computation $\lambda = s_0 \stackrel{g_1}{\to} s_1 \stackrel{g_2}{\to} s_2 \ldots$ can then be represented as the ω-word $(s_0 \stackrel{g_1}{\to} s_1) \cdot (s_1 \stackrel{g_2}{\to} s_2) \cdot (s_2 \stackrel{g_3}{\to} s_3) \cdots$ in Σ_M^ω. Given an infinite branch of the proof in $\mathcal{A}_{M,A,s^\star}$ corresponding to the extended

computation $s_0 \xrightarrow{\vec{u}_0} \langle s_0, f_0\rangle \to \langle g_1, s_1\rangle \xrightarrow{\vec{u}_1} \langle s_1, f_1\rangle \to \ \cdots$, its Σ_M-*projection* is defined as the sequence $(s_0 \xrightarrow{g_1} s_1)\cdot(s_1 \xrightarrow{g_2} s_2)\cdot(s_2 \xrightarrow{g_3} s_3)\cdots$ Formal correspondences between M and $\mathcal{A}_{M,A,s^\star}$ are below. It is worth noting that in the proofs of $\mathcal{A}_{M,A,s^\star}$, along all branches, there is a strict alternation between update rules and fork rules.

Proposition 1. *[12, Lemma 4] Let* $\mathrm{L} \subseteq \Sigma_M^\omega$ *and* $\vec{b} \in \mathbb{N}^r$*. There is a* $\vec{b}$*-strategy* F_A *w.r.t.* $s^\star$ *in* M *such that the set* $\mathtt{Comp}(s^\star, F_A)$ *of computations is included in* L *iff there is a proof in* $\mathcal{A}_{M,A,s^\star}$ *with root labelled by* $\langle s^\star, \vec{b}\rangle$*, every maximal branch being infinite, and its* Σ_M*-projection being in* L*.*

By way of example, let S_ϕ be the set of states in S satisfying the (state) formula ϕ (S is from some RB-CGS M) and L_ϕ be the restriction of Σ_M^ω to ω-words involving only states from S_ϕ (i.e., $s \xrightarrow{g} s'$ is allowed with $s, s' \in S_\phi$). By Proposition 1, $(M, s^\star) \models \langle\langle A^{\vec{b}}\rangle\rangle\, \mathsf{G}\,\phi$ iff there is a proof in $\mathcal{A}_{M,A,s^\star}$ restricted to locations involving only states in S_ϕ whose root is labelled by $\langle s^\star, \vec{b}\rangle$ and every maximal branch is infinite. This property can actually be generalised to any language in Σ_M^ω defined from a language in S^ω and this is used in the proof of forthcoming Lemma 7.

4.2.2. Carefully constructing AVASS

In this section, we explain how to handle the verification of $(M, s^\star) \models \langle\langle A^{\vec{b}}\rangle\rangle\Psi$, for $A \subseteq Ag$ and $\vec{b} \in \mathbb{N}^r$. As in Section 3, we assume w.l.o.g. that Ψ is a Boolean formula in NNF built over atomic formulae of the form $\mathsf{X}p$ or $p\,\mathsf{U}\,q$, where p, q are propositional variables. These are the building blocks to handle the general case, that is considered at a later stage (see Theorem 13). To determine the satisfaction of $(M, s^\star) \models \langle\langle A^{\vec{b}}\rangle\rangle\Psi$, we construct an AVASS $\mathcal{A}^\star = \langle Q^\star, r, R_1^\star, R_2^\star\rangle$, a colouring $\mathtt{col}^\star : Q^\star \to [0, 1]$ (defining a co-Büchi condition, as there are two colors and the smallest one is even), and $q^\star \in Q^\star$ such that

1. $(M, s^\star) \models \langle\langle A^{\vec{b}}\rangle\rangle\Psi$ iff there is a proof with root labelled by $\langle q^\star, \vec{b}\rangle$ such that all the maximal branches are infinite and the maximal (only) colour that appears infinitely often along each branch is 0.
2. $|Q^\star|$ is (only) exponential in $|M| + |\Psi|$.
3. $||R_1^\star|| \leq |Ag| \times \max(\{||wf(s, a, \mathtt{a})|| : s \in S, a \in Ag, \mathtt{a} \in Act\})$.

In Section 4.2.1, we have shown how to define an AVASS $\mathcal{A}_{M,A,s^\star} = \langle Q, r, R_1, R_2\rangle$ given M, A and $s^\star$. The AVASS $\mathcal{A}^\star$ is made of copies of $\mathcal{A}_{M,A,s^\star}$ so that the general behavior of $\mathcal{A}_{M,A,s^\star}$ is preserved, but we decorate the locations with a finite memory taking care of the satisfaction of the path formula Ψ.

Let us provide a few more definitions. Without loss of generality, we assume that Ψ is a positive Boolean combination of atomic path formulae of one of the following forms: $\mathsf{G}\,\ell$, $\ell_1\,\mathsf{U}\,\ell_2$, $\ell\,\mathsf{U}(\ell_1 \wedge \ell_2)$, $\mathsf{X}\,\ell$ where the ℓ's are literals. Notice that the transformation to obtain this shape can be performed in linear time (see Lemma 5).

Let $s \in S$ be a state and Φ be a Boolean combination of atomic path formulae of the form above. We write $s(\Phi)$ to denote the Boolean combinations of atomic path formulae obtained from Φ according to the following clauses:

1. If $\mathsf{G}\,\ell$ occurs in Φ and $(M,s) \not\models \ell$, then replace every occurrence of $\mathsf{G}\,\ell$ in Φ by $\bot$.
2. If $\ell_1 \,\mathsf{U}\, \ell_2$ occurs in Φ and $(M,s) \models \ell_2$ (resp. $(M,s) \not\models \ell_2 \vee \ell_1$) , then replace every occurrence of $\ell_1 \,\mathsf{U}\, \ell_2$ in Φ by $\top$ (resp. by $\bot$).
3. The clause for handling $\ell \,\mathsf{U}(\ell_1 \wedge \ell_2)$ is similar to (2).
4. If $\mathsf{X}\,\ell$ occurs in Φ and $(M,s) \models \ell$ (resp. $(M,s) \not\models \ell$), then replace every occurrence of $\mathsf{X}\,\ell$ in Ψ by $\top$ (resp. by $\bot$).

We write $s[\Phi]$ to denote the path formula defined as for $s(\Phi)$ except that only the first three clauses are considered (next-time atomic formulae remain untouched, if any). Intuitively, when Φ is a path formula to be satisfied on the extension of a current finite computation and s is visited next, the path formula $s(\Phi)$ is obtained from Φ by replacing the atomic path subformulae in Φ that are definitely true or false by $\top$ or $\bot$, respectively. The final value for $s(\Phi)$ is obtained by using simplification rules to eliminate or propagate $\top$ or $\bot$, if possible (e.g., $\bot \vee \phi$ is reduced to ϕ). Hence, $s(\Phi)$ may take the values $\bot$ or $\top$ (not strictly speaking path formulae, but we assume so below). The same simplifications are used for $s[\Phi]$. For technical reasons, we have also introduced the path formula $s[\Phi]$ that is computed without taking into account the next-time subformulae. This is helpful, when a path formula Φ is updated at the first position of a computation. Indeed, suppose that $(M,\lambda) \models \mathsf{X}\,\ell$ with $\lambda[0] = s$. Obviously, $(M,\lambda_{\geq 1}) \models \ell$ and $\lambda[1]$ satisfies the literal ℓ. In particular, if Φ contains $\mathsf{X}\,\ell$ and $(M,\lambda) \models \Phi$, then we may wish to replace $\mathsf{X}\,\ell$ by $\top$ in Φ but this can be done only after visiting $\lambda[1]$, whence the slight difference between the definitions of $s(\Phi)$ and $s[\Phi]$. We write $S(\Psi)$ to denote the set of path formulae obtained from Ψ by successive applications (possibly zero) of $s(\cdot)$ and $s[\cdot]$ for some $s \in S$, i.e.,

$$S(\Psi) \stackrel{\text{def}}{=} \{s_1(s_2(\cdots s_n[\Psi]\cdots)) \mid s_1,\ldots,s_n \in S, n \geq 0\}.$$

Observe that only the first application $s_n[\Psi]$ uses the transformation of the form $s[\cdot]$ and therefore ignores the X-formulae for the reasons just explained above. Here are its main properties.

Lemma 6. *Let Ψ be a path formula as above, M be an RB-CGS, and λ be one of its computations.*

(correctness.1) $(M,\lambda) \models \Psi$ *iff* $(M,\lambda) \models s[\Psi]$ *for* $\lambda[0] = s$.

(correctness.2) *If Ψ has no next-time atomic formulae, then* $(M,\lambda) \models s(\Psi)$ *iff* $(M,\lambda_{\geq 1}) \models s(\Psi)$ *for* $\lambda[0] = s$.

(correctness.3) $(M,\lambda) \models s[\Psi]$ *iff* $(M,\lambda_{\geq 1}) \models s'(s[\Psi])$ *for* $\lambda[0] = s$ *and* $\lambda[1] = s'$.

(small size) $\Psi \in S(\Psi)$ *and* $|S(\Psi)|$ *is in* $\mathcal{O}(3^{|\Psi|})$.

(stabilisation) *There is $i \geq 0$ such that for all $j \geq i$, we have* $\lambda[j](\cdots(\lambda[0][\Psi])\cdots) = \lambda[i](\cdots(\lambda[0][\Psi])\cdots)$.

As a consequence of Lemma 6, for all $i \geq 0$, we have $(M,\lambda) \models \Psi$ iff $(M,\lambda_{\geq i}) \models \lambda[i](\cdots(\lambda[0][\Psi])\cdots)$. By Lemma 6**(stabilisation)**, there exists a "limit" path formula $\Phi_{\lambda,\Psi}$ equal to some path formula of the form $\lambda[I](\cdots(\lambda[0][\Psi])\cdots)$ for some $I \geq 1$, such

that the satisfaction of a subset $\{\mathsf{G}\,\ell_1, \ldots, \mathsf{G}\,\ell_q\}$ of always formulae from $\Phi_{\lambda,\Psi}$ makes the path formula propositionally true, assuming that $(M, \lambda) \models \Psi$ holds true. In particular, this means that $(M, \lambda) \models \mathsf{G}\,\ell_1 \wedge \cdots \wedge \mathsf{G}\,\ell_q$. Without loss of generality, we can also asssume no position after I witnesses the satisfaction of an until formula from Φ (since there are a finite amount of positive occurrences of until formulae in Φ).

Proof. **(correctness.1)**

- If $\mathsf{G}\,\ell$ occurs in Ψ and $(M, s) \not\models \ell$, then $(M, \lambda) \not\models \mathsf{G}\,\ell$. Consequently, $(M, \lambda) \models \Psi$ iff $(M, \lambda) \models \Psi[\mathsf{G}\,\ell \leftarrow \bot]$.
- Similarly, if $\ell_1\,\mathsf{U}\,\ell_2$ occurs in Ψ and $(M, s) \models \ell_2$, then $(M, \lambda) \models \ell_1\,\mathsf{U}\,\ell_2$. Consequently, $(M, \lambda) \models \Psi$ iff $(M, \lambda) \models \Psi[\ell_1\,\mathsf{U}\,\ell_2 \leftarrow \top]$. As in the definition, taking care of formulae $\ell\,\mathsf{U}(\ell_1 \wedge \ell_2)$ is similar.
- If $\ell_1\,\mathsf{U}\,\ell_2$ occurs in Ψ and $(M, s) \not\models \ell_1 \vee \ell_2$, then $(M, \lambda) \not\models \ell_1\,\mathsf{U}\,\ell_2$. Consequently, $(M, \lambda) \models \Psi$ iff $(M, \lambda) \models \Psi[\ell_1\,\mathsf{U}\,\ell_2 \leftarrow \bot]$.

So let Φ be the path formula obtained from Ψ by applications of the above-mentioned substitutions, after applying the simplifications related to $\bot$ and $\top$. Note that the order in which the substitutions are applied is irrelevant as well as how the simplications are performed. The path formula Φ is actually equal to $s[\Psi]$. From the above properties, we conclude $(M, \lambda) \models \Psi$ iff $(M, \lambda) \models \Phi$.

(correctness.2) The proof is similar to the proof for **(correctness.1)** by observing that if the truth status of an until formula (resp. the falsity of an always formula) could not be determined at state s and Ψ has no next-time atomic formulae, then $(M, \lambda) \models \Psi$ iff $(M, \lambda_{\geq 1}) \models \Psi$.

(correctness.3) Suppose that $\mathsf{X}\,\ell$ occurs in Ψ.

- If $(M, s') \models \ell$, then $(M, \lambda) \models s[\Psi]$ iff $(M, \lambda) \models (s[\Psi])[\mathsf{X}\,\ell \leftarrow \top]$.
- If $(M, s') \not\models \ell$, then $(M, \lambda) \models s[\Psi]$ iff $(M, \lambda) \models (s[\Psi])[\mathsf{X}\,\ell \leftarrow \bot]$.

So let Φ be the path formula obtained from Ψ by applications of the above-mentioned substitutions, after applying the simplifications related to $\bot$ and $\top$. Again, note that the order in which the substitutions are applied is irrelevant as well as how the simplifications are performed thanks to *confluence*. Indeed, when Φ' (resp. Φ'') can be obtained from Φ by applying either a transformation from clauses (1)-(4) in the definition of $s(\cdot)$ or a simplication, then there is Ψ_C that can be obtained from Φ' (resp. Φ'') in at most one step, whence the confluence property. From the above properties, we have $(M, \lambda) \models \Psi$ iff $(M, \lambda) \models \Phi$.

As Φ has no next-time atomic formulae, by **(correctness.2)**, we have $(M, \lambda) \models \Phi$ iff $(M, \lambda_{\geq 1}) \models \Phi$. By **(correctness.1)**, we have $(M, \lambda_{\geq 1}) \models \Phi$ iff $(M, \lambda_{\geq 1}) \models s'(\Phi)$ ($s'(\Phi) = s'[\Phi]$ because Φ has no next-time X) and one can check that $s'(\Phi)$ is logically equivalent to $s'(s[\Psi])$.

(small size) By definition, $\Psi \in S(\Psi)$, as Ψ can be viewed as obtained from Ψ with no application of $s(\cdot)$ or $s[\cdot]$. Each original subformula of Ψ evolves in $\Phi \in S(\Psi)$ in three possible ways: (1) it remains as it is; (2) it is substituted by $\bot$; or (3) it is substituted by $\top$ (actually simplifications are also performed when $\bot \vee \ldots$ or $\top \wedge \ldots$ occur), whence $|S(\Psi)|$ is in $\mathcal{O}(3^{|\Psi|})$.
(stabilisation) As the size of $s(\Psi)$ is at most the size of Ψ, for any sequence $s_1, s_2, \ldots$, the size of $s_{n+1}(s_n(\cdots s_1[\Psi]\cdots))$ is at most the size of $s_n(s_{n-1}(\cdots s_1[\Psi]\cdots))$. Necessarily, the size of $s_n(s_{n-1}(\cdots s_1[\Psi]\cdots))$ stabilises from some position i. As any transformation strictly decreases the number of literals, we get the stabilisation of the path formula. □

Let us explain how $Q^\star$ is defined and used. By definition, $Q^\star$ is equal to $S(\Psi) \times Q$ so that each $\Phi \in S(\Psi)$ has its own copy of $\mathcal{A}_{M,A,s^\star}$. A location $\langle \Phi, q\rangle$ is intended to follow the rules of $\mathcal{A}_{M,A,s^\star}$ to satisfy the path formula Φ. In order to update Φ, when the proof visits a next location q' attached to the state $s' \in S$, the next location in $Q^\star$ becomes $\langle s'(\Phi), q'\rangle$. The (initial) location $q^\star$ is defined as the pair $\langle s^\star[\Psi], s^\star\rangle$, where $s^\star[\Psi]$ is built from the first three clauses only for defining $s(\cdot)$ (i.e., the atomic path formulae of the form $\mathsf{X}\,\ell$ are exceptionally ignored to start with), which is safe by Lemma 6**(correctness.1)**. Let us explain now how to define the update and fork rules when the locations in $S(\Psi) \times Q$ are involved ($\Phi \in S(\Psi)$).

- If $\langle s^\star, \vec{u}, \langle s^\star, f\rangle\rangle \in R_1$, then $\langle\langle \Phi, s^\star\rangle, \vec{u}, \langle \Phi, \langle s^\star, f\rangle\rangle\rangle \in R_1^\star$ and if $\langle\langle g, s\rangle, \vec{u}, \langle s, f\rangle\rangle \in R_1$, then $\langle\langle \Phi, \langle g, s\rangle\rangle, \vec{u}, \langle \Phi, \langle s, f\rangle\rangle\rangle \in R_1^\star$. For every $\Phi \in S(\Psi)$, the update rules (related to the actions of coalition A) do not modify the first argument Φ as no next location is reached yet.
- If $\langle\langle s, f\rangle, \langle g_1, s_1\rangle, \ldots, \langle g_\alpha, s_\alpha\rangle\rangle \in R_2$, then

$$\langle\langle \Phi, \langle s, f\rangle\rangle, \langle s_1(\Phi), \langle g_1, s_1\rangle\rangle, \ldots, \langle s_\alpha(\Phi), \langle g_\alpha, s_\alpha\rangle\rangle\rangle \in R_2^\star.$$

  Note that we move from Φ to $s_i(\Phi)$ in order to simplify Φ based on literals holding at s_i. We perform such an update for all the branches.

Consequently, a proof involving locations in $S(\Psi)\times Q$ leads to a proof in $\mathcal{A}_{M,A,s^\star}$ when the first component Φ in $\langle \Phi, q\rangle$ is ignored. For all $\langle \Phi, q\rangle$ in $Q^\star$, we have $\mathtt{col}^\star(\langle \Phi, q\rangle) \stackrel{\text{def}}{=} 0$ if $\{\mathsf{G}\,\ell_1, \ldots, \mathsf{G}\,\ell_n\} \models_{\text{PROP}} \Phi$, where $\mathsf{G}\,\ell_1$, …, $\mathsf{G}\,\ell_n$ are the G-formulae in Φ and $\models_{\text{PROP}}$ is the propositional entailment (which can be checked in LOGSPACE [68]). Otherwise, $\mathtt{col}^\star(\langle \Phi, q\rangle) \stackrel{\text{def}}{=} 1$. So for acceptance, we want to use locations $\langle \Phi, q\rangle$ with colour zero to be the only ones to occur infinitely often (and actually to stabilise on such a path formula Φ). In particular, by definition, $\mathtt{col}^\star(\langle \bot, q\rangle) \stackrel{\text{def}}{=} 1$ and $\mathtt{col}^\star(\langle \top, q\rangle) \stackrel{\text{def}}{=} 0$. The rationale for the definition of $\mathtt{col}^\star$ is simply that an infinite branch of a derivation in $\mathcal{A}^\star$, at some point Φ stabilises and all the states in S attached to the locations on that branch satisfy all the G-formulae in Φ (otherwise some of them would be replaced by $\bot$). The colour zero is attached only to locations in $Q^\star$ for which the satisfaction of all the G-formulae entails the satisfaction of the final path formula Φ, see also the paragraph after the presentation of Lemma 6. Correctness of the construction is stated below.

Lemma 7. *Let $\vec{b} \in \mathbb{N}^r$. Then, $(M, s^\star) \models \langle\!\langle A^{\vec{b}}\rangle\!\rangle \Psi$ iff there is a proof in $\mathcal{A}^\star$ whose root is labelled by $\langle q^\star, \vec{b}\rangle$ and all the maximal branches are infinite and the maximal colour that appears infinitely often is 0. Moreover, $(M, s^\star) \models \langle\!\langle A^{\vec{b}}\rangle\!\rangle \Psi$ can be checked in time exponential in $|M| + |\langle\!\langle A^{\vec{b}}\rangle\!\rangle \Psi|$.*

Proof. The proof combines advantageously the properties of $\mathcal{A}_{M,A,s^\star}$ [12] and the ones for $\mathcal{A}^\star$ related to the satisfaction of path formulae.

Let us start by performing the complexity analysis before providing the correctness arguments. As $|Q|$ is in $\mathcal{O}(|M|^2)$, we have $|Q^\star|$ is in $\mathcal{O}(3^{|\Psi|} \times |M|^2)$, see Lemma 6**(small size)**. The value $||R_1^\star||$ is bounded by $|Ag| \times \max(\{||wf(s,a,\mathtt{a})|| : s \in S, a \in Ag, \mathtt{a} \in Act\})$, which is bounded by $|M|2^{|M|}$ (as a direct consequence of the construction of $\mathcal{A}_{M,A,s^\star}$). By using the expression

$$(|Q| \times ||R_1||)^{2^{\mathcal{O}(r \times log(r+\mathtt{p}))}} + \mathcal{O}(r \times log||\vec{b}||)$$

for the parity game problem for AVASS (see Section 4.2.1), when r is fixed and $\mathtt{p} = 2$, $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$ can be checked in time exponential in $|M| + |\langle\langle A^{\vec{b}} \rangle\rangle \Psi|$.

Let $\vec{b} \in \mathbb{N}^r$. We write $\mathcal{A}_{M,A,s^\star} = \langle Q, r, R_1, R_2 \rangle$ to denote the AVASS defined in Section 4.2.1 given M, A and $s^\star$. Similarly, we write $\mathcal{A}^\star = \langle Q^\star, r, R_1^\star, R_2^\star \rangle$ to denote the AVASS defined from $\mathcal{A}_{M,A,s^\star}$ and Ψ as shown above. The locations in $Q^\star$ have one of the following forms:

$$\langle \Phi, s^\star \rangle, \qquad \langle \Phi, \langle s, f \rangle \rangle, \qquad \langle \Phi, \langle g, s \rangle \rangle,$$

where Φ is a path formula obtained from Ψ by replacing some atomic path formulae by either $\top$ or $\bot$, whereas $s^\star$, $\langle s, f \rangle$ and $\langle g, s \rangle$ are locations from Q. By construction of $\mathcal{A}^\star$, removing from a proof in $\mathcal{A}^\star$ the value Φ (projection on the second component of the locations), leads to a proof in $\mathcal{A}_{M,A,s^\star}$.

($\Rightarrow$) First, suppose that $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$. Without loss of generality, we assume that Ψ is a positive Boolean combination of atomic path formulae of one of the following forms: $\mathsf{G}\,\ell$, $\ell_1 \,\mathsf{U}\, \ell_2$, $\ell \,\mathsf{U} (\ell_1 \wedge \ell_2)$, $\mathsf{X}\,\ell$ where the ℓ's are literals. By definition of the satisfaction relation, there is a $\vec{b}$-strategy F_A with respect to $s^\star$ such that for all $\lambda \in \mathtt{Comp}(s^\star, F_A)$, we have $(M, \lambda) \models \Psi$. As done in [12, Section 4], the computations in $\mathtt{Comp}(s^\star, F_A)$ can be organised as an infinite tree corresponding to a derivation skeleton for $\mathcal{A}_{M,A,s^\star}$. Let us recall the definition of such a tree T_{F_A} equipped with a labelling function $\mathfrak{L} : T_{F_A} \to S$ and with a partial map $\mathfrak{R} : T_{F_A} \times T_{F_A} \to (\bigcup_{s' \in S} D_{Ag}(s'))$. We reproduce below some material from [12, Section 4] to have a self-contained formalisation. Below, we use some notations a bit different from those in the previous sections.

- $\mathfrak{L}(\varepsilon) \stackrel{\text{def}}{=} s^\star$ where ε is the root of T_{F_A}.
- For all finite words $w = k_1 \cdots k_\beta$ in T_{F_A} such that $\mathfrak{L}(w)$ is already defined, we add to T_{F_A} the values $k_1 \cdots k_\beta \cdot 1, \ldots, k_1 \cdots k_\beta \cdot \alpha$ such that
  - $F_A(\mathfrak{L}(\varepsilon) \xrightarrow{\mathfrak{R}(\varepsilon, k_1)} \mathfrak{L}(k_1) \xrightarrow{\mathfrak{R}(k_1, k_1k_2)} \mathfrak{L}(k_1k_2) \cdots \xrightarrow{\mathfrak{R}(k_1 \cdots k_{\beta-1}, k_1 \cdots k_\beta)} \mathfrak{L}(w)) = f$
  - $\{\langle g_1, s_1 \rangle, \ldots, \langle g_\alpha, s_\alpha \rangle\} = \{\langle g, s'' \rangle \in D_{Ag}(s') \times S \mid s'' = \delta(s', g), g \in D_{Ag}(s'), f \sqsubseteq g\}$ with $s' = \mathfrak{L}(w)$.
  - For all $j \in [1, \alpha]$, $\mathfrak{L}(w \cdot j) \stackrel{\text{def}}{=} s_j$ and $\mathfrak{R}(w, w \cdot j) \stackrel{\text{def}}{=} g_j$.

The tree T_{F_A} is defined by saturation of the above rules, and the maps $\mathfrak{L}$ and $\mathfrak{R}$ are defined accordingly. A maximal branch w of $\langle T_{F_A}, \mathfrak{R}, \mathfrak{L} \rangle$ is understood as an element of $(\mathbb{N} \setminus \{0\})^\omega$, such that any (strict) finite prefix of w belongs to T_{F_A}. The *label* of w, written $\mathtt{lab}(w)$, is defined as follows:

$$\mathfrak{L}(\varepsilon) \xrightarrow{\mathfrak{R}(\varepsilon, k_1)} \mathfrak{L}(k_1) \xrightarrow{\mathfrak{R}(k_1, k_1k_2)} \mathfrak{L}(k_1k_2) \cdots \xrightarrow{\mathfrak{R}(k_1 \cdots k_{\beta-1}, k_1 \cdots k_\beta)} \mathfrak{L}(k_1 \cdots k_\beta) \cdots$$

where $w = k_1k_2k_3\cdots$. By construction, $\mathtt{lab}(w)$ is a maximal computation (respecting the strategy F_A). The following properties are shown in [12, Section 4] and simply state that $\langle T_{F_A}, \mathfrak{R}, \mathfrak{L}\rangle$ contains all the computations from $s^\star$ that respect the strategy F_A.

1. **Completeness.** For every maximal computation λ starting at $s^\star$ and respecting F_A, there is a maximal branch w in $\langle T_{F_A}, \mathfrak{R}, \mathfrak{L}\rangle$ such that $\lambda = \mathtt{lab}(w)$.
2. **Soundness.** For every maximal branch w in $\langle T_{F_A}, \mathfrak{R}, \mathfrak{L}\rangle$, there is a maximal computation λ starting at $s^\star$ and respecting F_A such that $\mathtt{lab}(w) = \lambda$.

Let $\hat{T}_{F_A}$ be the tree obtained from T_{F_A} such that $k_1\cdots k_\beta$ in T_{F_A} implies $1\cdot k_1\cdots 1\cdot k_\beta$ in $\hat{T}_{F_A}$. The essential step now is to build a derivation skeleton $\mathcal{D} : \hat{T}_{F_A} \to (R_1 \cup R_2)$ as follows, where all the maximal branches of T_{F_A} are infinite.

- $\mathcal{D}(\varepsilon) = \langle s_0, wf_A(s_0, F_A(s_0)), \langle s_0, F_A(s_0)\rangle\rangle$ with $s_0 = s^\star$.
- $\mathcal{D}(1) = \langle\langle s_0, F_A(s_0)\rangle, \langle g_1, s_1\rangle, \ldots, \langle g_\alpha, s_\alpha\rangle\rangle$, where $1, \ldots, \alpha \in T_{F_A}$ (but $\alpha{+}1 \notin T_{F_A}$), and for all $j \in [1, \alpha]$, $\mathfrak{L}(j) = s_j$ and $\mathfrak{R}(\varepsilon, j) = g_j$. By construction of $\mathcal{A}_{M,A,s^\star}$, the rule $\mathcal{D}(1)$ is the unique fork rule starting from $\langle s_0, F_A(s_0)\rangle$.
- Let $\mathfrak{n} = 1k_11\cdots 1k_\beta$. By construction we can assume that we already have that $\mathcal{D}(1k_11\cdots k_{\beta-1}1) = \langle\langle s', f'\rangle, \langle g_1, s_1\rangle, \ldots, \langle g_\alpha, s_\alpha\rangle\rangle$.

  Let g' be equal to $\mathfrak{R}(k_1\cdots k_\beta, k_1\cdots k_\beta\cdot 1)$ and f be the restriction of g' to A (so $f \sqsubseteq g'$). Then,
  $$\mathcal{D}(\mathfrak{n}) = \langle\langle g_{k_\beta}, s_{k_\beta}\rangle, wf_A(s_{k_\beta}, f), \langle s_{k_\beta}, f\rangle\rangle.$$
- Let $\mathfrak{n} = 1k_11\cdots 1k_\beta 1$ with $k_1, \ldots, k_\beta \geq 1$ be such that
  $$\mathcal{D}(1k_11\cdots k_\beta) = \langle\langle g, s'\rangle, wf_A(s', f), \langle s', f\rangle\rangle \in R_1.$$
  Then, $\mathcal{D}(\mathfrak{n}) = \langle\langle s', f\rangle, \langle g_1, s_1\rangle, \ldots, \langle g_\alpha, s_\alpha\rangle\rangle$ where $k_1\cdots k_\beta 1, \ldots, k_1\cdots k_\beta\alpha \in T_{F_A}$ (but $k_1\cdots k_\beta(\alpha+1) \notin T_{F_A}$), and for all $j \in [1, \alpha]$,
  - $\mathfrak{L}(k_1\cdots k_\beta\cdot j) = s_j$ with $\beta \geq 1$;
  - $\mathfrak{R}(k_1\cdots k_\beta, k_1\cdots k_\beta j) = g_j$ with $\beta \geq 1$ and $1 \leq k_\beta \leq \alpha$.

  By construction of $\mathcal{A}_{M,A,s^\star}$, $\mathcal{D}(\mathfrak{n})$ is the unique fork rule starting from $\langle s', f\rangle$.

Given an infinite branch w of $\mathcal{D}$ (resp. w of the derivation $\hat{\mathcal{D}}$ based on $\mathcal{D}$), say $w = 1k_11k_21k_3\cdots \in \mathbb{N}^\omega$, we define the extended computation $\mathtt{ext}(w, A)$ as follows. Suppose that the label of such a branch is characterised by the values below. Any prefix of the form $1k_11\cdots k_\beta$ is associated with an update rule whereas any prefix of the form $1k_11\cdots k_\beta 1$ is associated with a fork rule (not necessarily binary). Intuitively, we specify for each position on the branch which rule is applied next.

$$\begin{array}{ll}
\mathcal{D}(\varepsilon) & = \langle s_0, \vec{u}_0, \langle s_0, f_0\rangle\rangle \\
\mathcal{D}(1) & = \langle\langle s_0, f_0\rangle, \langle g_1^1, s_1^1\rangle, \ldots, \langle g_{\alpha_1}^1, s_{\alpha_1}^1\rangle\rangle \\
\vdots & \\
\mathcal{D}(1k_11\cdots k_i) & = \langle\langle g_{k_i}^i, s_{k_i}^i\rangle, \vec{u}_{k_i}^i, \langle s_{k_i}^i, f_i\rangle\rangle. \\
\mathcal{D}(1k_11\cdots k_i1) & = \langle\langle s_{k_i}^i, f_i\rangle, \langle g_1^{i+1}, s_1^{i+1}\rangle, \ldots, \langle g_{\alpha_{i+1}}^{i+1}, s_{\alpha_{i+1}}^{i+1}\rangle\rangle \\
\vdots &
\end{array}$$

Then,

$$\texttt{ext}(w,A) \stackrel{\text{def}}{=} s_0 \stackrel{\vec{u}_0}{\to} \langle s_0, f_0\rangle \to \langle g^1_{k_1}, s^1_{k_1}\rangle \stackrel{\vec{u}_1}{\to} \langle s^1_{k_1}, f_1\rangle \to \langle g^2_{k_2}, s^2_{k_2}\rangle \stackrel{\vec{u}_2}{\to} \langle s^2_{k_2}, f_2\rangle \to \langle g^3_{k_3}, s^3_{k_3}\rangle \ \cdots$$

In [12, Lemma 3], the following properties are established.

(†) **Completeness.** For every maximal computation λ starting at $s^\star$ and respecting F_A, there is a maximal branch w in $\mathcal{D}$ such that $\texttt{ext}(\lambda, A) = \texttt{ext}(w, A)$. Note that $\texttt{ext}(\lambda, A)$ is defined earlier (see Section 4.2.1).

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

In order to define a proof $\mathcal{D}^\star$ in $\mathcal{A}^\star$ whose root is labelled by $\langle q^\star, \vec{b}\rangle$ such that all the maximal branches are infinite and the maximal colour that appears infinitely often is 0, it suffices to start from the proof $\hat{\mathcal{D}}$ and to decorate with path formulae $\Phi \in S(\Psi)$ each node of $\mathcal{D}$ (already labelled by a location q). Given $\Phi \in S(\Psi)$ and a computation λ, by Lemma 6**(correction.2)** we have $(M, \lambda) \models \Phi$ implies $(M, \lambda_{\geq 1}) \models s(\Phi)$ when $s = \lambda[0]$ and Φ has no next-time atomic formulae. Consequently, for each computation λ, we can define an ω-sequence $\Phi_0, \Phi_1, \ldots \in S(\Psi)^\omega$ such that for all $i \geq 0$, $\Phi_i = \lambda[i](\cdots(\lambda[0][\Psi])\cdots)$. Moreover, there is $I_\lambda \geq 0$ such that $\Phi_{I_\lambda}, \Phi_{I_\lambda+1}, \ldots \in \{\Phi_{I_\lambda}\}^\omega$ because of Lemma 6**(stabilisation)**.

Let w be a maximal branch in $\mathcal{D}$ such that $\texttt{ext}(w, A) = s_0 \stackrel{\vec{u}_0}{\to} \langle s_0, f_0\rangle \to \langle g^1_{k_1}, s^1_{k_1}\rangle \stackrel{\vec{u}_1}{\to} \langle s^1_{k_1}, f_1\rangle \to \langle g^2_{k_2}, s^2_{k_2}\rangle \stackrel{\vec{u}_2}{\to} \langle s^2_{k_2}, f_2\rangle \to \langle g^3_{k_3}, s^3_{k_3}\rangle \ \cdots$. By (‡) above, there is a maximal computation λ starting at $s^\star$ and respecting F_A such that $\texttt{ext}(w, A) = \texttt{ext}(\lambda, A)$. Let $\Phi_0, \Phi_1, \ldots \in S(\Psi)^\omega$ be the ω-sequence defined from λ (since it satisfies Ψ, a property inherited from F_A). Let w be the corresponding maximal branch in $\mathcal{D}^\star$ where the decoration is performed as follows (overloaded notation):

$$\begin{array}{rcl} \texttt{ext}(w, A) & = & \langle \Phi_0, s_0\rangle \stackrel{\vec{u}_0}{\to} \langle \Phi_0, \langle s_0, f_0\rangle\rangle \to \langle \Phi_1, \langle g^1_{k_1}, s^1_{k_1}\rangle\rangle \stackrel{\vec{u}_1}{\to} \\ & & \langle \Phi_1, \langle s^1_{k_1}, f_1\rangle\rangle \to \langle \Phi_2, \langle g^2_{k_2}, s^2_{k_2}\rangle\rangle \stackrel{\vec{u}_2}{\to} \ \langle \Phi_2, \langle s^2_{k_2}, f_2\rangle\rangle \to \ \cdots \end{array}$$

Note that for all $i \geq 0$, the path formula Φ_i appears in two consecutive locations (and this can happen many times if $\Phi_i = \Phi_j$ for some $j > i$). One can check that the colour zero occurs infinitely often along w because Φ_i stabilises to a value (say Φ_{I_λ}) such that all the always formulae hold true and make Φ_{I_λ} true propositionally. Actually, this is a consequence of $(M, \lambda) \models \Psi$ and the way the path formulae Φ_i's are computed (see Lemma 6).

($\Leftarrow$) Reciprocally, assume that there is a proof $\mathcal{D}^\star$ whose root is labelled by $\langle q^\star, \vec{b}\rangle$ such that all the maximal branches are infinite and the maximal colour that appears infinitely often is 0. We recall that Σ_M is the finite alphabet made of triples $\langle s', g, s''\rangle$ such that $\delta(s', g) = s''$ in M. Let L_Ψ be the subset of Σ_M^ω made of ω-sequences (LTL models) satisfying the path formula Ψ. By the bookkeeping of path formulae in $S(\Psi)$, one can establish that the Σ_M-projection of every maximal branch belongs to L_Ψ. Indeed, suppose that on the branch the projection of $S(\Psi)$ is the ω-sequence $\Psi_0, \Psi_1, \ldots$. We have $\Psi_0 = s[\Psi]$ where s is $s^\star$ and there is $t \geq 1$ such that for all $t' \geq t$, we have $\Psi_{t'} = \Psi_t$.

Finally, from the coloring map, we know that the satisfaction of always formulae of the form $\mathsf{G}\,\ell$ in Ψ_t makes it propositionally true. Moreover, all the underlying visited states satisfy ℓ.

By Proposition 1, there is a $\vec{b}$-strategy F_A w.r.t. $s^\star$ in M such that the set of computations $\mathtt{Comp}(s^\star, F_A)$ is included in L_Ψ. Consequently, there is a $\vec{b}$-strategy F_A w.r.t. $s^\star$ in M such that for all $\lambda \in \mathtt{Comp}(s^\star, F_A)$, we have $(M, \lambda) \models \Psi$. That is, $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$. □

4.2.3. The model-checking algorithm

We immediately prove the main result of the section.

Theorem 13. *For all $r \geq 1$,* $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(r))$ *is in* EXPTIME.

Proof. The proof of Theorem 13 makes use of a labelling algorithm, very similarly to the one in the proof of Theorem 11. However, to decide whether $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$, for $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$, we first perform a reduction of the dimension by identifying the component in $\vec{b}$ equal to ω. Then, we take advantage of Lemma 7 in order to check an instance of the corresponding parity game problem for AVASS, which can be done in exponential time. As the number of such requests is only polynomial in the size of the input formula, the whole algorithm runs in exponential time.

The general structure of the proof follows faithfully the (standard) structure of the proof for Theorem 11. Let $M = \langle Ag, r, S, Act, act, wf, \delta, L \rangle$ be an RB-CGS, and ϕ be a formula in $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$. Let us present Algorithm 2, an exponential-time labelling algorithm that computes the finite set $\{s \in S \mid (M, s) \models \phi\}$.

The algorithm works on the structure of formula ϕ. The cases for atomic formulae and Boolean connectives are immediate. Then, we consider in detail the case of strategic formulae $\langle\langle A^{\vec{b}} \rangle\rangle \Psi$. Without loss of generality, we assume that Ψ is a Boolean formula in NNF built over atomic path formulae of the form $\mathsf{X}\,\phi$ or $\phi\,\mathsf{U}\,\phi'$, where ϕ, ϕ' are state formulae. Let us assume that the maximal state formulae occurring in Ψ are $\phi_1, \ldots, \phi_N$ and the model-checking algorithm has already determined that they hold true exactly on the states in $S_1^\star, \ldots, S_N^\star$, respectively. Let $\Psi^\star$ be the formula obtained from Ψ by replacing each ϕ_i by a fresh atom p_i, and $M^\star$ be the RB-CGS obtained from M by modifying the labelling function as in the proof of Theorem 11: $L^\star(p_i) \stackrel{\text{def}}{=} S_i^\star$.

Concerning the case $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$, for $A \neq \emptyset$, we have $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$ iff $(M^\star, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi^\star$. Let $I \subseteq [1, r]$ be the set of components such that $\vec{b}(i) = \omega$ for all $i \in I$. We have $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$ iff $(M^{\star \setminus I}, s^\star) \models \langle\langle A^{\vec{b} \setminus I} \rangle\rangle \Psi^\star$ where $M^{\star \setminus I}$ (resp. $\vec{b} \setminus I$) is obtained from $M^\star$ (resp. $\vec{b}$) by removing the components in I. In Algorithm 2, we define $\mathcal{A}^\star = \langle Q^\star, r, R_1^\star, R_2^\star \rangle$, $\mathtt{col}^\star : Q^\star \to [0, 1]$, and a distinguished location $q^\star \in Q^\star$ as explained in Section 4.2.2, but built on $M^{\star \setminus I}$, $\Psi^\star$ and $s^\star$ instead. By Lemma 7, $(M, s^\star) \models \langle\langle A^{\vec{b}} \rangle\rangle \Psi$ iff there is a proof whose root is $\langle q^\star, \vec{b} \setminus I \rangle$ such that all the maximal branches are infinite and the maximal colour that appears infinitely often is 0.

Furthermore, note that we have the following equivalence: $(M, s^\star) \models \langle\langle \emptyset^{\vec{b}} \rangle\rangle \Psi$ iff $(K_{M^\star}, s^\star) \models \mathsf{A}\Psi^\star$ in CTL^+. It is worth noting that strictly speaking, the distinction between $\langle\langle \emptyset^{\vec{b}} \rangle\rangle$ and $\langle\langle A^{\vec{b}} \rangle\rangle$ in Algorithm 2 is not needed because the model-checking instance with CTL^+ can be formulated as an instance of the parity game problem for

AVASS. However, it allows us to pinpoint when a call to a subroutine about AVASS is really needed and when a routine about CTL^+ suffices.

Algorithm 2 – $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r)$ model checking –

1: **procedure** $\mathrm{MC}(M, \phi)$
2: **case** ϕ **of**
3: p: **return** $\{s \in S \mid s \in L(p)\}$
4: $\neg\psi$: **return** $S \setminus \mathrm{MC}(M, \psi)$
5: $\phi_1 \wedge \phi_2$: **return** $\mathrm{MC}(M, \phi_1) \cap \mathrm{MC}(M, \phi_2)$
6: $\langle\langle \emptyset^{\vec{b}} \rangle\rangle \Psi$: **return** $\{s \mid K_{M^\star}, s \models \mathsf{A}\Psi^\star \text{ in } \mathsf{CTL}^+\}$
7: $\langle\langle A^{\vec{b}} \rangle\rangle \Psi$: **return** $\{s^\star \in S \mid \mathcal{A}^\star, \mathtt{col}^\star, q^\star, \vec{b}^{\setminus I}$ is a positive instance of the parity game problem for AVASS $\}$
8: **end case**
9: **end procedure**

By structural induction, one can show that $(M, s) \models \phi$ iff $s \in \mathrm{MC}(M, \phi)$ (as in the proof of Theorem 11). As far as computational complexity is concerned, $\mathrm{MC}(M, \phi)$ is computed with a recursion depth linear in the size of ϕ and a polynomial amount of EXPTIME requests. The EXPTIME upper bound is due to Lemma 7 and the complexity analysis following the statement of Lemma 7. Note also that for each occurrence of a subformula ψ of ϕ, $\mathrm{MC}(M, \psi)$ can be computed only once, which guarantees the overall number of calls of the form $\mathrm{MC}(M, \psi)$: again, it is sufficient to take advantage of dynamic programming and to work with a table to remember the values $\mathrm{MC}(M, \psi)$ already computed (omitted in the present algorithm and already explained earlier). We also take advantage of the fact that the model-checking problem for CTL^+ is in $\Delta_2^{\mathrm{P}} \subseteq$ EXPTIME. □

Because of the results in [21, 12], we obtain immediately the following corollary.

Corollary 1. *Let $r \geq 4$ and $|Ag| \geq 2$. Then,* $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag, r))$ *is* EXPTIME*-complete*

The EXPTIME lower bound follows from [12, Corollary 1]. Specifically, in the proof of [12, Theorem 3] (by reduction from the control-state reachability for AVASS [21]) RB-CGS can be restricted to two agents. The EXPTIME upper bound is from Theorem 13. In the case of one resource, we obtain the following bounds.

Theorem 14. *For $r \in \{1, 2, 3\}$,* $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(r))$ *is* PSPACE*-hard and in* EXPTIME*.*

The PSPACE lower bound follows from the PSPACE-hardness of $\mathrm{MC}(\mathsf{ATL}^+)$ [27].

Discussion. Though establishing an EXPTIME upper bound for $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(1))$ reveals a substantial improvement in complexity compared to the 2EXPTIME-completeness of $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^*)$ [12], the exact complexity remains open and seems quite challenging. It is still unclear to us whether the developments in [27, Section 3.2] or in [28] about $\mathrm{MC}(\mathsf{ATL}^+)$ could be adapted to obtain an optimal upper bound.

As far as the complexity of $\mathrm{MC}(\mathsf{RB}{\pm}\mathsf{ATL}^+(1))$ is concerned, it is natural to wonder whether the model-checking problem for Parity-Energy ATL, also called pe-ATL in [69]

when the energy bound is bounded below and unbounded above can be used to analyse the complexity of MC($\mathsf{RB}{\pm}\mathsf{ATL}^+(1)$). MC($\mathsf{pe}$-$\mathsf{ATL}$) is in Δ_2^{P} [69] and it seems hopeless to take advantage of this bound[1] to solve efficiently MC($\mathsf{RB}{\pm}\mathsf{ATL}^+(1)$) since MC($\mathsf{ATL}^+$) is already PSPACE-hard [27]. A notable difference between $\mathsf{RB}{\pm}\mathsf{ATL}^+(1)$ and pe-ATL, is that the parity condition in pe-ATL is actually a global fairness condition for all the computations of the CGS.

5. Further Results: Applications and Extensions

In this section, we leverage on the results from the previous sections to derive interesting consequences for syntactic fragments of our languages (typically by bounding the number of linear-time temporal operators), as well as for logics that happen to be closely related to $\mathsf{RB}{\pm}\mathsf{ATL}$, $\mathsf{RB}{\pm}\mathsf{ATL}^+$, and $\mathsf{RB}{\pm}\mathsf{ATL}^*$.

5.1. Restricting the linear-time temporal operators

In this section, we introduce fragments of $\mathsf{RB}{\pm}\mathsf{ATL}^+$ obtained by restricting the set of path formulae by allowing only a finite amount of patterns for each fragment. This is a standard approach in the literature about temporal logics, as it allows us to restrict ourselves to a bounded number of temporal patterns supposedly useful in practice, in order to obtain more tractability results [24]. For instance, $\mathsf{RB}{\pm}\mathsf{ATL}$ can be understood as the fragment of $\mathsf{RB}{\pm}\mathsf{ATL}^+$ in which the only temporal patterns are $\mathsf{X}\,p$, $p_1\,\mathsf{U}\,p_2$ and $\mathsf{G}\,p$. For each fragment (defined below), we establish that its model-checking problem restricted to a single agent and a single resource can be solved in PTIME by building upon our algorithm to solve $\mathsf{RB}{\pm}\mathsf{ATL}^+(\{1\},1)$.

We recall that the path formulae for $\mathsf{RB}{\pm}\mathsf{ATL}^+(Ag,r)$ are obtained from the following grammar: $\Psi ::= \neg\Psi \mid \Psi \wedge \Psi \mid \mathsf{X}\,\phi \mid \phi\,\mathsf{U}\,\phi$. Given a finite set $\mathcal{O} = \{\oplus_1,\ldots,\oplus_n\}$ of *patterns* understood as built-in linear-time temporal operators, we associate to each pattern $\oplus_i$ an *arity* $a_i \geq 1$ and an LTL formula $\Psi_i(p_1,\ldots,p_{a_i})$ of temporal depth one (its *definition* that unambiguously provides a semantics to the operator $\oplus_i$). Hence, the formula $\Psi_i(p_1,\ldots,p_{a_i})$ is an LTL formula built from the grammar below, where $p,q \in \mathrm{AP}$:

$$\Psi ::= p \mid \neg\Psi \mid \Psi \wedge \Psi \mid p\,\mathsf{U}\,q \mid \mathsf{X}\,p,$$

We write $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}$ to denote the fragment of $\mathsf{RB}{\pm}\mathsf{ATL}^+$ in which path formulae are defined from the grammar below:

$$\Psi ::= \oplus_1(\phi_1,\ldots,\phi_{a_1}) \mid \ldots \mid \oplus_n(\phi_1,\ldots,\phi_{a_n}),$$

where the ϕ_i's are state formulae and $\mathcal{O} = \{\oplus_1,\ldots,\oplus_n\}$.

Let $\mathtt{t}$ be the map that transforms $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}$ formulae into $\mathsf{RB}{\pm}\mathsf{ATL}^+$ formulae using the definitions for the linear-time temporal operators in $\mathcal{O}$. Namely, $\mathtt{t}$ is the identity for propositional variables and it is homomorphic for Boolean connectives. Finally, for all $\oplus_i \in \mathcal{O}$, we have $\mathtt{t}(\oplus_i(\phi_1,\ldots,\phi_{a_i})) \stackrel{\text{def}}{=} \Psi_i(\mathtt{t}(\phi_1),\ldots,\mathtt{t}(\phi_{a_i}))$. The satisfaction relation $\models$ for $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}$ is essentially defined as for $\mathsf{RB}{\pm}\mathsf{ATL}^+$ except for formulae of the

[1] We thank Dario Della Monica for checking that the NP upper bound stated in [69, Theorem 5.1] is actually an Δ_2^{P} upper bound in view of the algorithm developped in [69, Section 4].

form $\oplus_i(\phi_1, \ldots, \phi_{a_i})$. More precisely, $(M, s) \models \phi$ in $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}} \stackrel{\text{def}}{\Leftrightarrow} (M, s) \models \mathtt{t}(\phi)$ in $\mathsf{RB}{\pm}\mathsf{ATL}^+$.

Let us explain how $\mathsf{RB}{\pm}\mathsf{ATL}$ can be viewed as a syntactic fragment of $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}^\star}$ for some finite set $\mathcal{O}^\star = \{\oplus_1, \oplus_2, \oplus_3\}$ of patterns, with $a_1 = 1$ and $a_2 = a_3 = 2$. We define $\Psi_1(p_1) \stackrel{\text{def}}{=} \mathsf{X}\, p_1$, $\Psi_2(p_1, p_2) \stackrel{\text{def}}{=} p_1 \,\mathsf{U}\, p_2$ and $\Psi_3(p_1, p_2) \stackrel{\text{def}}{=} \neg(p_1 \,\mathsf{U}\, p_2)$. The definitions related to $\oplus_1$ and $\oplus_2$ are well-motivated to capture the formulae of the form $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \mathsf{X}\, \phi$ and $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \phi_1 \,\mathsf{U}\, \phi_2$ from $\mathsf{RB}{\pm}\mathsf{ATL}$, respectively. However, the pattern $\oplus_3$ is instrumental to capture formulae of the form $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \mathsf{G}\, \phi$. Indeed, in LTL, $\mathsf{G}\, p$ is logically equivalent to $\neg(\top \,\mathsf{U}\, \neg p)$ and therefore $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \mathsf{G}\, p$ from $\mathsf{RB}{\pm}\mathsf{ATL}$ is logically equivalent to $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \oplus_3(\top, \neg p)$ in $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}^\star}$. It is even immediate to add the release operator R, not necessarily native to $\mathsf{RB}{\pm}\mathsf{ATL}$, as $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \phi_1 \,\mathsf{R}\, \phi_2$ is captured by $\langle\!\langle A^{\vec{b}} \rangle\!\rangle \oplus_3 (\neg\phi_1, \neg\phi_2)$ in $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}^\star}$ (recall that state formulae are closed under negation, unlike path formulae in $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}^\star}$).

The main result of interest herein about $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}$ is stated below (with associated arities and definitions). Observe that $\mathsf{RB}{\pm}\mathsf{ATL}$ is equal to $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}$ for some $\mathcal{O}$.

Theorem 15. *Let $\mathcal{O} = \{\oplus_1, \ldots, \oplus_n\}$ be a finite fixed set of patterns. The model-checking problem for $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}(\{1\}, 1)$ is in* PTIME.

Proof. Let $\mathcal{O} = \{\oplus_1, \ldots, \oplus_n\}$ be a fixed finite set of patterns (with associated definitions $\Psi_1, \ldots, \Psi_n$ and the corresponding map $\mathtt{t}$ parameterised by $\mathcal{O}$). The PTIME upper bound is obtained by running exactly the algorithm described in the proof of Theorem 11 and by taking advantage of the fact that $\mathcal{O}$ is fixed in $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}(\{1\}, 1)$ so that the model-checking algorithm runs in PTIME.

To this end, let us first establish a few properties about $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}(\{1\}, 1)$. For all $i \in [1, n]$, the LTL formula $\Psi_i(p_1, \ldots, p_{a_i})$ is a constant of the logic $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}(\{1\}, 1)$. Further, $\Psi_i(p_1, \ldots, p_{a_i})$ can be put in NNF in linear-time in the size of $\Psi_i(p_1, \ldots, p_{a_i})$; and since $\neg\mathsf{X}\, p$ is logically equivalent to $\mathsf{X}\neg p$ and $\neg(p \,\mathsf{U}\, p')$ is logically equivalent to $\mathsf{G}\neg p' \vee (\neg p' \,\mathsf{U}\, (\neg p \wedge \neg p'))$, $\Psi_i(p_1, \ldots, p_{a_i})$ is logically equivalent to a path formula $\Psi'_i(p_1, \ldots, p_{a_i})$ in NNF such that the negation $\neg$ occurs only in front of propositional variables, and the atomic formulae have one of the forms $\mathsf{X}\, C$, $\mathsf{G}\, C$, and $C \,\mathsf{U}\, C'$, with C, C' being conjunctions of literals. In particular, $\Psi'_i(p_1, \ldots, p_{a_i})$ can be computed in linear-time in the size of $\Psi_i(p_1, \ldots, p_{a_i})$, also by using the fact that X and G distributes over $\wedge$. As in the proof of Lemma 5, guessing which atomic formulae in Ψ'_i hold to evaluate Ψ'_i to true, amounts to guess a conjunction of the form

$$\mathsf{X}\, C_0 \wedge \mathsf{G}\, C'_0 \wedge (C_1 \,\mathsf{U}\, C'_1) \wedge \cdots \wedge (C_m \,\mathsf{U}\, C'_m),$$

for some $m \in \mathbb{N}$, where the C_i's and C'_j's are arbitrary conjunctions of literals (empty conjunctions are equivalent to $\top$). Each conjunction above can be understood as a propositional valuation on the path formula $\Psi'_i(p_1, \ldots, p_{a_i})$, and because $\Psi'_i(p_1, \ldots, p_{a_i})$ is fixed, the number of relevant conjunctions that need to be considered is exponential in the size of Ψ_i but, more importantly, it is a constant for the logic $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}(\{1\}, 1)$. Hence, as explained in the proof of Lemma 5, checking whether $(M, s) \models \langle\!\langle \{1\}^b \rangle\!\rangle \oplus_i(\phi_1, \ldots, \phi_{a_i})$ holds amounts to invoke a constant number of instances of GREACH(1-VASS). This constant number depends on the number of valuations/conjunctions satisfying $\Psi'_i(p_1, \ldots, p_{a_i})$ and on the constant number of guesses (see again the proof of Lemma 5) to construct the

instances of GREACH(1-VASS). Hence, in the labelling algorithm from the proof of Theorem 11, the case for formulae of the form $\langle\langle\{1\}^b\rangle\rangle \oplus_i (\phi_1, \ldots, \phi_{a_i})$ requires polynomial time only (assuming that $\mathcal{O}$ is fixed), which allows us to get an overall complexity in PTIME. □

As a corollary, we obtain the main result established in [39].

Corollary 2. *Model-checking* $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$ *is in* PTIME.

Obviously, if $\mathcal{O}$ is expressive enough for $\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}$ to embed CTL, both problems MC($\mathsf{RB}{\pm}\mathsf{ATL}^+{}_{\mathcal{O}}(\{1\},1)$) and MC($\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$) are PTIME-complete since PTIME-hardness is inherited from the model-checking problem for CTL [43]. Note also that PTIME fragments of ATL^+ have been investigated in [70].

5.2. Model-checking $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$ *is in* PTIME*: a refined algorithm*

In Section 5.1, we showed that model-checking $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$ is in PTIME using instances of the decision problem GREACH(1-VASS). Below, we provide a different polynomial-time algorithm for solving MC($\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$) in which we only use instances of CREACH(1-VASS) and NONTER(1-VASS) (instead of the more general decision problem GREACH(1-VASS), which is not necessary for $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$). The interest of this different proof derives from the particular version of the problem MC($\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$) we consider in this section, which differs in two points from the problem analysed in Section 5.1. The first point concerns the use of the linear-time temporal operator release R instead of the always operator G, which provides more generality in the syntax. The second point provides some restriction as we assume that in the input RB-CGS, there is some idle action `idle` that can be triggered from any state and for any agent and such that its cost is zero. The latter hypothesis has been assumed in [39], following developments in [12], and it happens to be essential to use instances of CREACH(1-VASS). The algorithm presented below appears in [39] and compared to the algorithm underlying the proof of Corollary 2, it is designed to be implemented more efficiently. Hereafter, for every $b \in \mathbb{N} \cup \{\omega\}$, we write $\langle\langle b\rangle\rangle\phi$ instead of $\langle\langle\{1\}^b\rangle\rangle\phi$.

As done in Section 4, given an RB-CGS $M = \langle\{1\}, S, Act, 1, act, wf, \delta, L\rangle$ with a single agent and a single resource, the corresponding 1-VASS $V_M = \langle S, 1, R_V\rangle$ is such that $q \xrightarrow{u} q' \in R_V$ iff there is some action $\mathtt{a} \in act(q,1)$ such that $\delta(q,\mathtt{a}) = q'$ and $wf(q,1,\mathtt{a}) = u$. Similarly, we write $K_M = \langle S, R, L_K\rangle$ to denote the Kripke structure such that qRq' iff there is some action $\mathtt{a} \in act(q,1)$ such that $\delta(q,\mathtt{a}) = q'$ and $L_K(q) = \{q\}$ (by a slight abuse of notations, we assume that $AP = Q$). Note that, thanks to the `idle` action, K_M is now a total Kripke structure, i.e., every world has at least one successor.

We now investigate the relationship between computations in M and runs in V_M and in K_M, respectively. Below, given a b-consistent computation $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$, the sequence of associated resource values is denoted by $(v_i)_{i\in\mathbb{N}}$. We recall that $v_0 = b$ and $v_{i+1} = v_i + wf(q_i, 1, \mathtt{a}_i)$ for all $i \geq 0$.

Lemma 8. *Let M be an RB-CGS with a single agent and a single resource.*

(I) *Let $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ be a b-consistent computation associated to the family $(v_i)_{i\in\mathbb{N}}$ of resource values. If $b \in \mathbb{N}$, then $\langle q_0, v_0\rangle \to \langle q_1, v_1\rangle \to \langle q_2, v_2\rangle \cdots$ is an infinite run in V_M; otherwise (i.e., $b = \omega$) $q_0 \to q_1 \to q_2 \cdots$ is an infinite path in K_M.*

(II) *Let* $\langle q_0, v_0\rangle \to \langle q_1, v_1\rangle \to \langle q_2, v_2\rangle \cdots$ *be an infinite run in* V_M*. Then, there is a* v_0*-consistent computation* $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ *associated to the family* $(v_i)_{i\in\mathbb{N}}$ *of resource values.*

(III) *Let* $q_0 \to q_1 \to q_2 \cdots$ *be an infinite path in* K_M*. Then, there is an* ω*-consistent computation* $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ *in* M*.*

Proof. (I) Let $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ be a b-consistent computation associated to the family $(v_i)_{i\in\mathbb{N}}$ of resource values, and $b \in \mathbb{N}$. We show that for all $i \geq 0$, $\langle q_i, v_i\rangle \xrightarrow{t} \langle q_{i+1}, v_{i+1}\rangle$ in V_M holds for some transition t in V_M. Indeed, by definition, for $i \geq 0$, $q_i \xrightarrow{\mathtt{a}_i} q_{i+1}$ in M iff $\delta(q_i, \mathtt{a}_i) = q_{i+1}$ for some $u = wf(q_i, 1, \mathtt{a})$. Then, by definition of V_M, we have that $q_i \xrightarrow{u} q_{i+1}$. Moreover, the resource availability $v_{i+1} = v_i + u$ is greater than zero, as the computation is b-consistent, and therefore $\langle q_i, v_i\rangle \xrightarrow{t} \langle q_{i+1}, v_{i+1}\rangle$ holds for $t = q_i \xrightarrow{u} q_{i+1}$. Clearly, if $b = \omega$, $q_0 \to q_1 \to q_2 \cdots$ is an infinite path in K_M.

(II) Let $\langle q_0, v_0\rangle \to \langle q_1, v_1\rangle \to \langle q_2, v_2\rangle \cdots$ be an infinite run in V_M. We show that in M, for all $i \geq 0$, $q_i \xrightarrow{\mathtt{a}_i} q_{i+1}$ and $v_i \geq 0$. Indeed, for $i \geq 0$, $\langle q_i, v_i\rangle \xrightarrow{t} \langle q_{i+1}, v_{i+1}\rangle$ holds iff for some transition $t = q_i \xrightarrow{u} q_{i+1}$ and $v_{i+1} = u + v_i \geq 0$. By definition of V_M, this means that in M there is some action $\mathtt{a}_i \in act(q_i, 1)$ such that $\delta(q_i, \mathtt{a}_i) = q_{i+1}$ and $wf(q_i, 1, \mathtt{a}) = u$. Hence, $q_i \xrightarrow{\mathtt{a}_i} q_{i+1}$ and $v_i \geq 0$. Therefore, $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ is a v_0-consistent computation associated to the family $(v_i)_{i\in\mathbb{N}}$ of resource values.

(III) If $q_0 \to q_1 \to q_2 \cdots$ is an infinite path in K_M. Then, by reasoning similarly to point (II) we can show that $q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$, where each action $\mathtt{a}_i$ is such that $\delta(q_i, \mathtt{a}_i) = q_{i+1}$, is an ω-consistent computation in M. □

Lemma 8 is instrumental to the three following lemmas that are at the heart of the model-checking algorithm for $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\}, 1)$. As earlier in the paper, given $S_1 \subseteq S$, we write $V_M^{S_1}$ (resp. $K_M^{S_1}$) to denote the restriction of V_M (resp. K_M) to the states in S_1 only.

Lemma 9. *Let* M *be an RB-CGS for* $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\}, 1)$*,* $S_1 \subseteq S$ *with* $s \in S_1$*, and* $b \in \mathbb{N}$*.*

(I) *There is a* b*-consistent computation starting at* s *in* M *that visits only states in* S_1 *iff* $(V_M^{S_1}, \langle s, b\rangle)$ *is a positive instance of NONTER(1-VASS).*

(II) *There is an* ω*-consistent computation starting in* s *in* M *that visits only states in* S_1 *iff* $(K_M, s) \models \mathsf{E}\,\mathsf{G}(\bigvee_{s'\in S_1} s')$ *in* CTL*.*

Proof. (I) ⇒ Suppose that $\lambda = q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ is a b-consistent computation starting in $s = q_0$ that visits only states in S_1. By Lemma 8.(I), $\langle q_0, v_0\rangle \to \langle q_1, v_1\rangle \to \langle q_2, v_2\rangle \cdots$ is an infinite run in $V_M^{S_1}$ for $v_0 = b$. As a result, $(V_M^{S_1}, \langle s, b\rangle)$ is a positive instance of NONTER(1-VASS). For the ⇐-direction the proof is similar, while using Lemma 8.(II).

(II) ⇒ Suppose that $\lambda = q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ is an ω-consistent computation starting in $s = q_0$ that visits only states in S_1. By Lemma 8.(I), $q_0 \to q_1 \to q_2 \cdots$ is an infinite path in K_M. Hence, the CTL formula $\mathsf{E}\,\mathsf{G}(\bigvee_{s'\in S_1} s')$ is true at s in K_M. For the ⇐-direction the proof is similar, while using Lemma 8.(III). □

Hence, Lemma 9 is a consequence of Lemma 8, which will be generalised in Lemma 11. Let us focus now on the until operator U.

Lemma 10. *Let M be an RB-CGS for $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$, $S_1, S_2 \subseteq S$ with $s \in S$, and $b \in \mathbb{N}$.*

(I) *There is a b-consistent computation starting at s in M such that its projection on S is in $S_1^* \cdot S_2 \cdot S^\omega$ (understood as an ω-regular expression) iff for some $s' \in S_2$, $(V_M^{S_1 \cup S_2}, \langle s, b\rangle, s')$ is a positive instance of CREACH(1-VASS).*

(II) *There is an ω-consistent computation starting at s in M such that its projection on S is in $S_1^* \cdot S_2 \cdot S^\omega$ iff in CTL, we have*

$$(K_M, s) \models \mathsf{E}(\bigvee_{s' \in S_1} s')\,\mathsf{U}(\bigvee_{s'' \in S_2} s'').$$

Proof. (I) $\Rightarrow$ Suppose that $\lambda = q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ is a b-consistent computation starting at s in M such that its projection on S is in $S_1^* \cdot S_2 \cdot S^\omega$. By Lemma 8.(I), $\langle q_0, v_0\rangle \to \langle q_1, v_1\rangle \to \langle q_2, v_2\rangle \cdots$ is an infinite run in V_M for $v_0 = b$, whose projection is in $S_1^* \cdot S_2 \cdot S^\omega$ and $q_n \in S_2$. Then, clearly $(V_M^{S_1 \cup S_2}, \langle s, b\rangle, q_n)$ is a positive instance of CREACH(1-VASS). For the $\Leftarrow$-direction the proof is similar, while using Lemma 8.(II) and the `idle` action that allows us to extend a finite trace to an infinite one.

(II) $\Rightarrow$ Suppose that $\lambda = q_0 \xrightarrow{\mathtt{a}_0} q_1 \xrightarrow{\mathtt{a}_1} q_2 \cdots$ is an ω-consistent computation starting in $s = q_0$ such that its projection on S is in $S_1^* \cdot S_2 \cdot S^\omega$. By Lemma 8.(I), $q_0 \to q_1 \to q_2 \cdots$ is an infinite path in K_M. Hence, the CTL formula $\mathsf{E}(\bigvee_{s' \in S_1} s')\,\mathsf{U}(\bigvee_{s'' \in S_2} s'')$ is true at s in K_M. For the $\Leftarrow$-direction the proof is similar, while using Lemma 8.(III). □

This is again a consequence of Lemma 8 but here, we have to use the fact that the distinguished action `idle` is enabled in any state (which is handy to extend to infinity a finite witness run). Finally, we consider the linear-time temporal operator R.

Lemma 11. *Let M be an RB-CGS for $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$, $S_1, S_2 \subseteq S$ with $s \in S$, and $b \in \mathbb{N}$.*

(I) *There is a b-consistent computation starting at s in M such that its projection on S is in $S_2^\omega \cup (S_2 \setminus S_1)^* \cdot (S_1 \cap S_2) \cdot S^\omega$ iff either $(V_M^{S_2}, \langle s, b\rangle)$ is a positive instance of NONTER(1-VASS) or for some $s' \in S_1 \cap S_2$, $(V_M^{S_2}, \langle s, b\rangle, s')$ is a positive instance of CREACH(1-VASS).*

(II) *There is an ω-consistent computation starting at s in M such that its projection on S is in $S_2^\omega \cup (S_2 \setminus S_1)^* \cdot (S_1 \cap S_2) \cdot S^\omega$ iff in CTL, we have*

$$(K_M, s) \models (\mathsf{E}\,\mathsf{G} \bigvee_{s'' \in S_2} s'') \vee \mathsf{E}(\bigvee_{s' \in (S_2 \setminus S_1)} s')\,\mathsf{U}(\bigvee_{s'' \in S_1 \cap S_2} s'').$$

CTL with the release operator R is known to also admit a PTIME model-checking problem (see, e.g., [24, Chapter 7]). The CTL formula in Lemma 11(II) can be equivalently replaced by

$$\mathsf{E}(\bigvee_{s' \in S_1} s')\,\mathsf{R}(\bigvee_{s'' \in S_2} s'').$$

In Algorithm 3 below, we shall use the release operator R in CTL in order to simplify the involved formulae.

Proof. (I) $\Rightarrow$ If there is a b-consistent computation λ starting in s that only visits states in S_2, then by Lemma 9.(I), $(V_M^{S_2}, \langle s, b\rangle)$ is a positive instance of NONTER(1-VASS). On the other hand, if the projection of λ on S belongs to $(S_2 \setminus S_1)^* \cdot (S_1 \cap S_2) \cdot S^\omega$, then by Lemma 10(I), for some $s' \in S_1 \cap S_2$, $(V_M^{(S_2 \setminus S_1) \cup (S_1 \cap S_2)}, \langle s, b\rangle, s')$ is a positive instance of CREACH(1-VASS). Since $(S_2 \setminus S_1) \cup (S_1 \cap S_2) = S_2$, $(V_M^{S_2}, \langle s, b\rangle, s')$ is also a positive instance of CREACH(1-VASS).

For the $\Leftarrow$-direction the proof is similar. In particular, if $(V_M^{S_2}, \langle s, b\rangle, s')$ is a positive instance of CREACH(1-VASS) for some $s' \in S_1 \cap S_2$, then by Lemma 10.(I), there is a b-consistent computation λ starting at s in M such that its projection on S is in $(S_2 \setminus S_1)^* \cdot (S_1 \cap S_2) \cdot S^\omega$. Similarly, if $(V_M^{S_2}, \langle s, b\rangle)$ is a positive instance of NONTER(1-VASS), by Lemma 9(I), there is a b-consistent computation λ starting at s in M such that its projection on S is in S_2^ω.

(II) The result follows similarly by item (II) in Lemma 9 and 10. □

Let $M = \langle \{1\}, S, Act, 1, act, wf, \delta, L\rangle$ be a resource-bounded CGS, and ϕ be a formula in $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\}, 1)$. Let us present below Algorithm 3 that is a polynomial-time algorithm that computes the finite set $GMC(M, \phi)$ (by default, $b \in \mathbb{N}$).

By induction, it can be shown that $(M, s) \models \phi$ iff $s \in GMC(M, \phi)$. Lemma 10 and Lemma 11 are used to prove the soundness of the subroutines for U and R, with $b \in \mathbb{N} \cup \{\omega\}$, respectively. When the strategy modality is $\langle\langle \emptyset^b \rangle\rangle$, it behaves as the standard path quantifier A in CTL^*, which is reflected in Algorithm 3. As far as computational complexity is concerned, $GMC(M, \phi)$ is computed with a recursion depth linear in the size of ϕ and the control-state reachability and nontermination problems can be solved in polynomial time by Theorem 7 and 8. More precisely, for each occurrence of a subformula ψ of ϕ, $GMC(M, \psi)$ can be computed only once, which guarantees the overall number of calls of the form $GMC(M, \psi)$ to be at most polynomial: it is sufficient to take advantage of dynamic programming and to work with a table to remember the values $GMC(M, \psi)$ already computed (omitted in the present algorithm, see also the proof of Theorem 11). It is also worth observing that the instances we consider are polynomial in the sizes of M and ϕ. Finally, we take advantage of the fact that the model-checking problem for CTL including R remains in PTIME (see, e.g., [24, Chapter 7]).

Consequently, reasoning about a single resource in CTL comes at no extra computational cost. Hence, in principle we can verify specification such as formula (1) in Example 2 efficiently.

*5.3. Comparison with RBTL and RBTL**

In this section we discuss the resource-bounded temporal logic RBTL^*, which extends CTL^* by adding resources [13, 38]. In particular, we show that this logic is essentially the same as single-agent $\mathsf{RB}{\pm}\mathsf{ATL}^*$. While such a result is not surprising, apparently it has so far been overlooked in the literature. Indeed, in [12], complexity results are given independently for both RBTL^* and single-agent $\mathsf{RB}{\pm}\mathsf{ATL}^*$, even though the two logics will be proved to be translatable one into the other. Such an equivalence allows us to apply results for single-agent $\mathsf{RB}{\pm}\mathsf{ATL}$ to RBTL as well. We first introduce the syntax and semantics of RBTL^* as given in [38].

Definition 16. *Given $r \geq 1$, the* state formulae ϕ *and* path formulae Ψ *in RBTL^* are*

Algorithm 3 – $\mathsf{RB}{\pm}\mathsf{ATL}(\{1\},1)$ Model-checking –

1: **procedure** GMC(M,ϕ)
2: **case** ϕ **of**
3: p: **return** $\{s \in S \mid s \in L(p)\}$
4: $\neg\psi$: **return** $S \setminus GMC(M,\psi)$
5: $\psi_1 \wedge \psi_2$: **return** $GMC(M,\psi_1) \cap GMC(M,\psi_2)$
6: $\langle\!\langle b \rangle\!\rangle \mathsf{X}\,\psi$: **return** $\{s \mid \exists\, \mathsf{a} \in act(s,1),\ 0 \le b + wf(s,1,\mathsf{a}),\ \delta(s,\mathsf{a}) \in GMC(M,\psi)\}$
7: $\langle\!\langle \omega \rangle\!\rangle \mathsf{X}\,\psi$: **return** $\{s \mid \exists\, \mathsf{a} \in act(s,1),\ \delta(s,\mathsf{a}) \in GMC(M,\psi)\}$
8: $\langle\!\langle \emptyset^{\omega} \rangle\!\rangle \mathsf{X}\,\psi$: **return** $\{s \mid \forall\, \mathsf{a} \in act(s,1),\ \delta(s,\mathsf{a}) \in GMC(M,\psi)\}$
9: $\langle\!\langle \emptyset^{b} \rangle\!\rangle \mathsf{X}\,\psi$: **return** $\{s \mid \forall\, \mathsf{a} \in act(s,1),\ \delta(s,\mathsf{a}) \in GMC(M,\psi)\}$
10: $\langle\!\langle b \rangle\!\rangle \psi_1\, \mathsf{U}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid \exists\, s' \in S_2$ s.t. $(V_M^{S_1 \cup S_2}, \langle s,b\rangle, s')$ is a positive
11: inst. of CREACH(1-VASS) $\}$
12: $\langle\!\langle \omega \rangle\!\rangle \psi_1\, \mathsf{U}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid K_M, s \models \mathsf{E}(\bigvee_{s' \in S_1} s')\, \mathsf{U}(\bigvee_{s' \in S_2} s')\}$
13: $\langle\!\langle \emptyset^{\omega} \rangle\!\rangle \psi_1\, \mathsf{U}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid K_M, s \models \mathsf{A}(\bigvee_{s' \in S_1} s')\, \mathsf{U}(\bigvee_{s' \in S_2} s')\}$
14: $\langle\!\langle \emptyset^{b} \rangle\!\rangle \psi_1\, \mathsf{U}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid K_M, s \models \mathsf{A}(\bigvee_{s' \in S_1} s')\, \mathsf{U}(\bigvee_{s' \in S_2} s')\}$
15: $\langle\!\langle b \rangle\!\rangle \psi_1\, \mathsf{R}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid (V_M^{S_2}, \langle s,b\rangle)$ is a positive inst. of NONTER(1-VASS) $\}$
$\cup \{s \in S \mid \exists\, s' \in S_1 \cap S_2$ s.t. $(V_M^{S_2}, \langle s,b\rangle, s')$ is a positive inst. of CREACH(1-VASS) $\}$
16: $\langle\!\langle \omega \rangle\!\rangle \psi_1\, \mathsf{R}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid K_M, s \models \mathsf{E}(\bigvee_{s' \in S_1} s')\, \mathsf{R}(\bigvee_{s' \in S_2} s')\}$
17: $\langle\!\langle \emptyset^{\omega} \rangle\!\rangle \psi_1\, \mathsf{R}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid K_M, s \models \mathsf{A}(\bigvee_{s' \in S_1} s')\, \mathsf{R}(\bigvee_{s' \in S_2} s')\}$
18: $\langle\!\langle \emptyset^{b} \rangle\!\rangle \psi_1\, \mathsf{R}\, \psi_2$: $S_1 := GMC(M,\psi_1)$; $S_2 := GMC(M,\psi_2)$;
return $\{s \in S \mid K_M, s \models \mathsf{A}(\bigvee_{s' \in S_1} s')\, \mathsf{R}(\bigvee_{s' \in S_2} s')\}$
19: **end case**
20: **end procedure**

built according to the following BNF grammar:

$$\begin{array}{lcl} \phi & ::= & p \mid \neg\phi \mid \phi \wedge \phi \mid \langle\vec{b}\rangle\Psi \\ \Psi & ::= & \phi \mid \neg\Psi \mid \Psi \wedge \Psi \mid \mathsf{X}\,\Psi \mid \Psi\,\mathsf{U}\,\Psi, \end{array}$$

where $p \in \mathrm{AP}$ *and* $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$.

Formulae in RBTL^* *are understood as the state formulae.*

The fragment RBTL of RBTL^* is obtained by restricting path formulae just like in the case of $\mathsf{RB}{\pm}\mathsf{ATL}$: $\Psi ::= \mathsf{X}\,\phi \mid \phi\,\mathsf{U}\,\phi \mid \mathsf{G}\,\phi$. The fragment RBTL^+ is defined from RBTL^* as $\mathsf{RB}{\pm}\mathsf{ATL}^+$ is defined from $\mathsf{RB}{\pm}\mathsf{ATL}^*$ in Section 2. In [12] the interpretation of RBTL^* is given on a particular class of models, based on vector addition systems with states.

Definition 17 (Model). *A model for* RBTL^* *is a tuple* $A = \langle Q, r, R, L\rangle$ *such that (i)* $\langle Q, r, R\rangle$ *is a* serial *VASS, i.e., for every* $q \in Q$*, there exist* $q' \in Q$ *and* $\vec{u} \in \mathbb{Z}^r$ *such that* $(q, \vec{u}, q') \in R$*; and (ii)* $L : AP \to \mathcal{P}(Q)$ *is a* labelling function.

The assumption of seriality on models appeared originally in [38], but it is omitted, for example, in [12]. Here we assume seriality to facilitate the comparison with $\mathsf{RB}{\pm}\mathsf{ATL}^*$, whose models are also serial by definition.

In a model A, a *pseudo-run* ρ is an infinite sequence $(q_0, \vec{v}_0) \to (q_1, \vec{v}_1) \to \ldots$ such that for all $i \geq 0$, there exists $(q, \vec{u}, q') \in R$ such that $q_i = q$, $q_{i+1} = q'$, and $\vec{v}_{i+1} = \vec{u} + \vec{v}_i$ (with $n + \omega = \omega$). A pseudo-run ρ is a *run* iff for all $i \geq 0$, $\vec{v}_i \in (\mathbb{N} \cup \{\omega\})^r$. This is consistent with the notions used so far.

Definition 18 (Satisfaction relation). *The satisfaction relation* $\models$ *in model* A*, for state* $q \in Q$*, run* ρ*,* $p \in \mathrm{AP}$*, state formula* ϕ*, and path formula* Ψ *is defined as follows (clauses for Boolean connectives are immediate and thus omitted):*

$$\begin{array}{ll} (A,q) \models p & \textit{iff } \ q \in L(p) \\ (A,q) \models \langle\vec{b}\rangle\Psi & \textit{iff for some infinite run } \rho \textit{ from } (q,\vec{b}),\ (A,\rho) \models \Psi \\ (A,\rho) \models \phi & \textit{iff } \ (A,\rho[0]) \models \phi \\ (A,\rho) \models \mathsf{X}\,\Psi & \textit{iff } \ (A,\rho_{\geq 1}) \models \Psi \\ (A,\rho) \models \Psi\,\mathsf{U}\,\Psi' & \textit{iff for some } i \geq 0,\ (A,\rho_{\geq i}) \models \Psi' \textit{, and for all } 0 \leq j < i,\ (A,\rho_{\geq j}) \models \Psi \end{array}$$

The model-checking problem for RBTL^* takes as inputs a model A, a location q and an RBTL^* state formula ϕ, and asks whether $(A, q) \models \phi$ holds. A similar definition can be provided for all relevant fragments of RBTL^*.

We now prove that the logics RBTL^* and $\mathsf{RB}{\pm}\mathsf{ATL}^*(\{1\}, r)$ are semantically equivalent, in the sense that truth-preserving translations exist between models and formulae. First, consider the translation maps τ and τ' between RBTL^* and $\mathsf{RB}{\pm}\mathsf{ATL}^*(\{1\}, r)$ such that τ, τ' are the identity on AP, they are homomorphic for Boolean and temporal operators, and $\tau(\langle\vec{b}\rangle\Psi) \stackrel{\text{def}}{=} \langle\!\langle \{1\}^{\vec{b}} \rangle\!\rangle\, \tau(\Psi)$, $\tau'(\langle\!\langle \{1\}^{\vec{b}} \rangle\!\rangle \Psi) \stackrel{\text{def}}{=} \langle\vec{b}\rangle\tau'(\Psi)$, and $\tau'(\langle\!\langle \emptyset^{\vec{b}} \rangle\!\rangle \Psi) \stackrel{\text{def}}{=} [\vec{\omega}]\tau'(\Psi)$, where $[\vec{\omega}]\Psi \stackrel{\text{def}}{=} \neg\langle\vec{\omega}\rangle\neg\Psi$.

Further, given an RB-CGS $M = \langle\{1\}, AP, S, Act, r, act, wf, \delta, L\rangle$ with a single agent 1, define the associated model $A_M = \langle V_M, L\rangle$ for RBTL^* such that V_M is the VASS associated to M, as defined in Section 4. Symmetrically, given a model $A = \langle Q, r, R, L\rangle$, define the associated single-agent RB-CGS $M_A = \langle\{1\}, r, Q, R, act, wf, \delta, L\rangle$ (the set of actions is equal to the set of transitions R) such that for every $q \in Q$,

- $act(q,1) = \{(q', \vec{u}, q'') \in R \mid q = q'\}$;
- for every $(q, \vec{u}, q') \in act(q,1)$, $wf(q, 1, (q, \vec{u}, q')) = \vec{u}$;
- for every $(q, \vec{u}, q') \in act(q,1)$, $\delta(q, (q, \vec{u}, q')) = q'$.

Notice that, by the assumption of seriality in Definition 17, for every $q \in Q$, $act(q,1) \neq \emptyset$, as required by Definition 3 of RB-CGS.

We now state the following auxiliary lemma, whose proof follows immediately by the definitions of A_M and M_A above.

Lemma 12. *1. Given a single-agent RB-CGS M, state $s \in S$, and budget $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$,*

(a) for every $\vec{b}$-consistent computation λ starting at s in M, in A_M there exists a run ρ_λ from $(s, \vec{b})$ such that for every $i \geq 0$, $(\lambda_i, \vec{v}_i) \to (\lambda_{i+1}, \vec{v}_{i+1})$ with $\vec{v}_{i+1} = \vec{v}_i + wf(\lambda_i, 1, \mathsf{a}_i)$ and $\lambda_i \xrightarrow{\mathsf{a}_i} \lambda_{i+1}$.

(b) for every run ρ from $(s, \vec{b})$ in A_M, $\rho = \rho_\lambda$ for some $\vec{b}$-consistent computation λ starting at s in M.

2. Given a model A for RBTL*, state $q \in Q$, and budget $\vec{b} \in (\mathbb{N} \cup \{\omega\})^r$,

(a) for every run ρ from $(q, \vec{b})$ in A, in M_A there exists a $\vec{b}$-consistent computation λ_ρ such that for every $i \geq 0$, $\lambda_\rho[i] \xrightarrow{(\rho_i, \vec{u}, \rho_{i+1})} \lambda_\rho[i+1]$ with $(\rho_i, \vec{v}_i) \to (\rho_{i+1}, \vec{v}_{i+1})$ and $\vec{v}_{i+1} = \vec{u} + \vec{v}_i$.

(b) for every $\vec{b}$-consistent computation λ starting at q in M_A, $\lambda = \lambda_\rho$ for some run ρ starting at $(q, \vec{b})$ in A.

Proof. (sketch) 1a) Let λ be a $\vec{b}$-consistent computation from $s \in S$, say $\lambda = s_0 \xrightarrow{\mathsf{a}_0} s_1 \xrightarrow{\mathsf{a}_1} s_2 \cdots$ with the associated resource values $(\vec{v}_i)_{i \in \mathbb{N}}$. We recall that $\vec{v}_0 = \vec{b}$ and $\vec{v}_{i+1} = \vec{v}_i + wf(s_i, 1, \mathsf{a}_i)$ for all $i \geq 0$. Let us consider the sequence $(s_0, \vec{v}_0) \to \cdots \to (s_i, \vec{v}_i) \cdots$. By definition of A_M, for all $i \geq 0$, we have $(s_i, \vec{v}_i) \xrightarrow{t_i} (s_{i+1}, \vec{v}_{i+1})$ in A_M with $t_i = s_i \xrightarrow{wf(s_i, 1, \mathsf{a}_i)} s_{i+1}$. Consequently, ρ is a run in A_M from the configuration $(s, \vec{b})$ as $s = s_0$ and $\vec{b} = \vec{v}_0$.

1b) Consider a run ρ from $(s, \vec{b})$ in A_M, and the $\vec{b}$-consistent computation $\lambda = s_0 \xrightarrow{\mathsf{a}_0} s_1 \xrightarrow{\mathsf{a}_1} s_2 \cdots$, with associated resource values $(\vec{v}_i)_{i \in \mathbb{N}}$, such that $s_0 = s$ and for every $i \geq 0$, $\rho_i = (s_i, \vec{v}_i)$ and $\vec{v}_{i+1} = \vec{v}_i + wf(s_i, 1, \mathsf{a}_i)$. It is not difficult to check that $\rho = \rho_\lambda$.

The proof for item (2) is similar. □

By using Lemma 12 we can finally prove that RB±ATL$^*(\{1\}, r)$ and RBTL* are closely related semantically.

Theorem 19.

1. For every state formula ϕ and path formula Ψ in RBTL*, and model A with state $q \in Q$ and run ρ,

$$(A, q) \models \phi \quad \textit{iff} \quad (M_A, q) \models \tau(\phi)$$
$$(A, \rho) \models \Psi \quad \textit{iff} \quad (M_A, \lambda_\rho) \models \tau(\Psi)$$

where λ_ρ is obtained from ρ as in Lemma 12.(2a).

2. *For every state formula ϕ' and path formula Ψ' in RB±ATL*, and single-agent RB-CGS M with state $s \in S$ and computation λ,*

$$(M, s) \models \phi' \quad \textit{iff} \quad (A_M, s) \models \tau'(\phi')$$
$$(M, \lambda) \models \Psi' \quad \textit{iff} \quad (A_M, \rho_\lambda) \models \tau'(\Psi')$$

where ρ_λ is obtained from λ as in Lemma 12.(1a).

Proof. Both items can be proved by mutual induction on the structure of formulae. The base cases for propositional variables and the inductive cases for Boolean and temporal operators are immediate.

Item (1), case for $\phi = \langle\vec{b}\rangle\Psi$. $\Rightarrow$ Suppose that $(A, q) \models \phi$, that is, for some run ρ from $(q, \vec{b})$, $(A, \rho) \models \Psi$. By Lemma 12.(2a), in M_A we can construct a $\vec{b}$-consistent computation λ_ρ from q such that $(M_A, \lambda_\rho) \models \tau(\Psi)$ by the induction hypothesis. This means that $(M_A, q) \models \langle\!\langle\{1\}^{\vec{b}}\rangle\!\rangle\ \tau(\Psi) = \tau(\langle\vec{b}\rangle\Psi) = \tau(\phi)$. $\Leftarrow$ If $(M_A, q) \models \langle\!\langle\{1\}^{\vec{b}}\rangle\!\rangle\tau(\Psi) = \tau(\phi)$, then for some $\vec{b}$-consistent computation λ, $(M_A, \lambda) \models \tau(\Psi)$. By Lemma 12.(2b), $\lambda = \lambda_\rho$ for some run ρ from $(q, \vec{b})$ in A, and by induction hypothesis we obtain $(A, \rho) \models \Psi$, that is, $(A, q) \models \phi$.

Item (2), case for $\phi' = \langle\!\langle\{1\}^{\vec{b}}\rangle\!\rangle\Psi'$. $\Rightarrow$ Suppose that $(M, s) \models \phi'$, that is, for some $\vec{b}$-consistent computation λ from s, $(M, \lambda) \models \Psi'$. By Lemma 12.(1a), in A_M we can construct a run ρ_λ from $(s, \vec{b})$ such that $(A_M, \rho_\lambda) \models \tau'(\Psi')$ by induction hypothesis. Hence, $(A_M, s) \models \langle\vec{b}\rangle\tau'(\Psi') = \tau'(\phi')$. $\Leftarrow$ Suppose that $(A_M, s) \models \tau'(\phi') = \langle\vec{b}\rangle\tau'(\Psi')$. That is, for some run ρ from $(s, \vec{b})$, $(A_M, \rho) \models \tau'(\Psi')$. By Lemma 12.(1b) we have that $\rho = \rho_\lambda$ for some $\vec{b}$-consistent computation λ such that $(M, \lambda) \models \Psi'$ by induction hypothesis. Hence, $(M, s) \models \phi'$.

Case for $\phi' = \langle\!\langle\emptyset^{\vec{b}}\rangle\!\rangle\Psi'$. $\Rightarrow$ Suppose that $(A_M, s) \not\models \tau'(\phi') = [\vec{\omega}]\tau'(\Psi')$. That is, for some run ρ from $(s, \vec{\omega})$, $(A_M, \rho) \not\models \tau'(\Psi')$. By Lemma 12.(1b) we have that $\rho = \rho_\lambda$ for some $\vec{\omega}$-consistent computation λ from s such that $(M, \lambda) \models \Psi'$ by induction hypothesis. Hence, $(M, s) \not\models \phi'$. $\Leftarrow$ Suppose that $(M, s) \not\models \phi'$, that is, for some computation λ from s, $(M, \lambda) \not\models \Psi'$. By Lemma 12.(1a), in A_M there exists a run ρ_λ from $(s, \vec{\omega})$ such that $(A_M, \rho_\lambda) \not\models \tau'(\Psi')$ by induction hypothesis. Hence, $(A_M, s) \not\models [\vec{\omega}]\tau'(\Psi') = \tau'(\phi')$. □

As a consequence of Theorem 19, RBTL* and the restriction of RB±ATL* to a single agent are essentially the same logic in the sense that their translations according to τ, τ' are semantically faithful when single-agent RB-CGS are understood as RBTL* models (i.e., a VASS with a valuation). A similar result holds for RBTL (resp. RBTL$^+$) and single-agent RB±ATL (resp. RB±ATL$^+$), as τ, τ' preserves these fragments. Based on the correspondences established in Theorem 19 and the fact that translations τ, τ' are polynomial, we derive the complexity results for the model-checking problem in Fig. 6. Notice that, for all numbers of resources, we have tight complexity bounds. Moreover, for the case of a single resource, the model-checking problem is no harder than for the corresponding fragment of CTL*. Thus, we conclude that reasoning about a single resource in CTL* and fragments comes at no extra computational cost.

6. Conclusions and Future Work

In this work, we have pushed further our understanding of the relationship between reasoning about resources in ATL-logics and decision problems for vector addition systems

r	RBTL	RBTL$^+$	RBTL*
∞	EXPSPACE-c. [12, Th. 4]		
$\geq$ **4**	in PSPACE [12, Cor. 2]		
3	PSPACE-h. [29] [27]		
2			
1	in PTIME **(Cor. 2)**	in Δ_2^{P} **(Th. 11)**	in PSPACE ([12, Cor. 2])
	PTIME-h. (from CTL)	Δ_2^{P}-h. (from CTL$^+$ [30])	PSPACE-h. (from CTL*)

Figure 6: The complexity of model-checking RBTL, RBTL$^+$, and RBTL*.

with states (VASS) or variants. This allowed us to establish new complexity results about model-checking problems for ATL-logics with resources and to identify several fragments with low complexity.

More precisely, we proved that the model-checking problem for RB$\pm$ATL$^+(\{1\},1)$ is in Δ_2^{P} (Theorem 11) by essentially introducing the generalised reachability problem for 1-VASS, and showing it to be in PTIME (Theorem 10). Hence, MC(RB$\pm$ATL$^+(\{1\},1)$) is no more complex than MC(CTL$^+$), despite the greater expressive power due to the presence of one resource. Additionally, we have established that the model-checking problem for RB$\pm$ATL$(\{1\},1)$ is PTIME-complete and actually this holds for any fragment of RB$\pm$ATL$^+(\{1\},1)$ with a fixed finite set of linear-time temporal operators (see Section 5.1). For $r \geq 1$, we proved MC(RB$\pm$ATL$^+(r)$) to be in EXPTIME, with an identical upper bound for its subproblem MC(RB$\pm$ATL(r)). To show this, we presented a reduction to the parity game problem for AVASS that is optimal complexity-wise, so that we avoid the doubly-exponential blow-up observed when defining the reduction from MC(RB$\pm$ATL*) in [12]. When $r \geq 4$, we obtain EXPTIME-completeness (Corollary 1).

Our main contributions are directed towards reducing the complexity gaps for many meaningful fragments of RB$\pm$ATL* and identifying fragments with a tractable model-checking problem. Apart from the application to practical case studies that would take advantage of our identified tractable fragments and associated algorithms, here are research directions for future work.

- What is the precise complexity of MC(RB$\pm$ATL$^+(Ag,1)$) with arbitrary finite sets Ag of agents? Currently, this problem is known to be in EXPTIME for $|Ag| \geq 2$ and PSPACE-hard from MC(ATL$^+$) [27]). However, it is an open problem whether there is a decision procedure running in polynomial space. To this end, refined analyses from [71, 72, 25] might help.

- It is well-known that CTL and CTL$^+$ are equally expressive [26], and this result extends to ATL and ATL$^+$ [27]. In both cases, the characterisation of the complexity for the respective model-checking problem differs. It is an open problem whether RB$\pm$ATL(Ag,r) and RB$\pm$ATL$^+(Ag,r)$, for $|Ag| \geq 2$ and $r \geq 1$, are equally expressive.

- Another challenging issue consists in characterising the complexity of the problem MC(RB$\pm$ATL$(Ag,1)$) for arbitrary finite sets of agents (currently known to be in PSPACE for $|Ag| \geq 2$ [19] and PTIME-hard from MC(CTL) [43]).

Acknowledgment. We would like to deeply thank the anonymous referees for all their suggestions that helped us a lot improve the quality of this document. We would like also to thank Michael Blondin (University of Sherbrooke) for pointing us to [31] and for insightful feedback, as well as the anonymous referees at AAMAS'19 and ECAI'20 for their suggestions and comments that helped us improve the conference papers [39] and [40]. Francesco Belardinelli acknowledges the support of the ANR JCJC Project SVeDaS (ANR-16-CE40- 0021) and Stéphane Demri acknowledges the support of the Centre National de la Recherche Scientifique (C.N.R.S.).

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

[2] R. Alur, L. de Alfaro, T. Henzinger, S. Krishnan, F. Mang, S. Qadeer, S. Rajamani, S. Tasiran, MOCHA user manual, Tech. rep., University of California at Berkeley (2000).

[3] F. Laroussinie, N. Markey, Augmenting ATL with strategy contexts, Information and Computation (245) (2015) 98–123.

[4] M. Pauly, A modal logic for coalitional power in games, Journal of Logic and Computation 12 (1) (2002) 149–166.

[5] K. Chatterjee, T. Henzinger, N. Piterman, Strategy logic, in: CONCUR'07, Vol. 4703 of Lecture Notes in Computer Science, Springer, 2007, pp. 59–73.

[6] F. Mogavero, A. Murano, G. Perelli, M. Vardi, Reasoning about strategies: On the model-checking problem, ACM Transactions on Computational Logic 15 (4) (2014) 34:1–34:47.

[7] P. Cermák, A. Lomuscio, F. Mogavero, A. Murano, MCMAS-SLK: A model checker for the verification of strategy logic specifications, in: CAV'14, Vol. 8559 of Lecture Notes in Computer Science, Springer, 2014, pp. 525–532.

[8] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: an open-source model checker for the verification of multi-agent systems, Software Tools for Technology Transfer 19 (1) (2017) 9–30.

[9] R. Alur, L. de Alfaro, R. Grosu, T. Henzinger, A. Thomas, M. Kang, C. Kirsch, R. M. F. Mang, B.-Y. Wang, jMocha: A model checking tool that exploits design structure, in: Proceedings of the 23rd International Conference on Software Engineering (ICSE01), IEEE, 2001, pp. 835–836.

[10] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, A. Zbrzezny, Verics 2007 - a model checker for knowledge and real-time, Fundamenta Informaticae 85 (1) (2008) 313–328.

[11] N. Alechina, B. Logan, State of the art in logics for verification of resource-bounded multi-agent systems, in: A. Blass, P. Cégielski, N. Dershowitz, M. Droste, B. Finkbeiner (Eds.), Fields of Logic and Computation III - Essays Dedicated to Yuri Gurevich on the Occasion of His 80th Birthday, Vol. 12180 of Lecture Notes in Computer Science, Springer, 2020, pp. 9–29.

[12] N. Alechina, N. Bulling, S. Demri, B. Logan, On the complexity of resource-bounded logics, Theoretical Computer Science 750 (2018) 69–100.

[13] N. Bulling, B. Farwer, On the (Un-)Decidability of Model-Checking Resource-Bounded Agents, in: ECAI'10, 2010, pp. 567–572.

[14] D. Della Monica, M. Napoli, M. Parente, On a logic for coalitional games with priced-resource agents, Electronic Notes in Theoretical Computer Science 278 (2011) 215–228.

[15] N. Bulling, V. Goranko, How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract), in: Proceedings 1st International Workshop on Strategic Reasoning (SR'13), Vol. 112 of EPTCS, 2013, pp. 33–41.

[16] N. Alechina, B. Logan, H. Nguyen, F. Raimondi, Decidable model-checking for a resource logic with production of resources, in: ECAI'14, 2014, pp. 9–14.

[17] N. Alechina, N. Bulling, B. Logan, H. 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.

[18] N. Alechina, B. Logan, H. Nguyen, A. Rakib, A logic for coalitions with bounded resources, in: IJCAI'09, 2009, pp. 659–664.

[19] N. Alechina, B. Logan, H. Nguyen, F. Raimondi, Model-checking for resource-bounded ATL with production and consumption of resources, Journal of Computer and System Sciences 88 (2017) 126–144.

[20] R. Karp, R. Miller, Parallel program schemata, Journal of Computer and System Sciences 3 (2) (1969) 147–195.

[21] J. Courtois, S. Schmitz, Alternating vector addition systems with states, in: MFCS'14, Vol. 8634 of Lecture Notes in Computer Science, Springer, 2014, pp. 220–231.
[22] M. Jurdziński, R. Lazić, S. Schmitz, Fixed-dimensional energy games are in pseudo-polynomial time, in: ICALP'15, Vol. 9135 of Lecture Notes in Computer Science, Springer, 2015, pp. 260–272.
[23] P. Abdulla, R. Mayr, A. Sangnier, J. Sproston, Solving Parity Games on Integer Vectors, in: CONCUR'13, Vol. 8052 of Lecture Notes in Computer Science, Springer, 2013, pp. 106–120.
[24] S. Demri, V. Goranko, M. Lange, Temporal Logics in Computer Science, Cambridge University Press, 2016.
[25] T. Colcombet, M. Jurdziński, R. Lazić, S. Schmitz, Perfect half space games, in: LiCS'17, IEEE Press, 2017, pp. 1–11.
[26] A. Emerson, J. Halpern, Decision procedures and expressiveness in the temporal logic of branching time, Journal of Computer and System Sciences 30 (1985) 1–24.
[27] N. Bulling, W. Jamroga, Verifying agents with memory is harder than it seemed, AI Communications 23 (4) (2010) 389–403.
[28] V. Goranko, A. Kuusisto, R. Rönnholm, Game-theoretic semantics for ATL^+ with applications to model checking, in: AAMAS'17, ACM, 2017, pp. 1277–1285.
[29] 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.
[30] F. Laroussinie, N. Markey, P. Schnoebelen, Model checking CTL^+ and FCTL is hard, in: FoSSaCS'01, Vol. 2030 of Lecture Notes in Computer Science, Springer, 2001, pp. 318–331.
[31] L. Rosier, H.-C. Yen, A multiparameter analysis of the boundedness problem for vector addition systems, Journal of Computer and System Sciences 32 (1986) 105–135.
[32] K. Chatterjee, L. Doyen, T. Henzinger, The cost of exactness in quantitative reachability, in: Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday, Vol. 10460 of Lecture Notes in Computer Science, Springer, 2017, pp. 367–381.
[33] J. Girard, Linear logic, Theoretical Computer Science (1987) 1–101.
[34] U. Engberg, G. Winskel, Completeness Results for Linear Logic on Petri Nets, Annals of Pure and Applied Logic 86 (2) (1997) 101–135.
[35] F. Paoli, Substructural Logics: A Primer, Vol. 13, 2002.
[36] N. Alechina, B. Logan, H. Nguyen, F. Raimondi, Symbolic model checking for one-resource RB+-ATL, in: IJCAI'15, AAAI Press, 2015, pp. 1069–1075.
[37] N. Alechina, N. Bulling, B. Logan, H. Nguyen, The virtues of idleness: A decidable fragment of resource agent logic, Artificial Intelligence 245 (2017) 56–85.
[38] N. Bulling, B. Farwer, Expressing properties of resource-bounded systems: The logics RBTL* and RBTL, in: CLIMA X, Vol. 6214 of Lecture Notes in Computer Science, Springer, 2009, pp. 22–45.
[39] F. Belardinelli, S. Demri, Resource-bounded ATL: the quest for tractable fragments, in: AAMAS'19, 2019, pp. 206–214.
[40] F. Belardinelli, S. Demri, Reasoning with a bounded number of resources in ATL+, in: ECAI'20, 2020, pp. 624–631.
[41] N. Alechina, B. Logan, H. Nguyen, F. Raimondi, L. Mostarda, Symbolic model-checking for resource-bounded ATL, in: AAMAS'15, International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1809–1810.
[42] N. Alechina, B. Logan, H. Nguyen, A. Rakib, Resource-bounded alternating-time temporal logic, in: AAMAS'10, IFAAMAS, 2010, pp. 481–488.
[43] P. Schnoebelen, The complexity of temporal logic model checking, in: AiML'02, Vol. 4 of Advances in Modal Logic, King's College Publications, 2003, pp. 437–459.
[44] P. Jančar, Z. Sawa, A note on emptiness for alternating finite automata with a one-letter alphabet, Information Processing Letters 104 (5) (2007) 164–167.
[45] D. Figueira, R. Lazić, J. Leroux, F. Mazowiecki, G. Sutre, Polynomial-space completeness of reachability for succinct branching VASS in dimension one, in: ICALP'17, Vol. 80 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017, pp. 119:1–119:14.
[46] S. Almagor, N. Cohen, G. Pérez, M. Shirmohammadi, J. Worrell, Coverability in 1-VASS with Disequality Tests, in: CONCUR'20, Vol. 285 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 38:1–38:20.
[47] E. Mayr, An algorithm for the general Petri net reachability problem, SIAM Journal of Computing 13 (3) (1984) 441–460.
[48] R. Kosaraju, Decidability of reachability in vector addition systems, in: STOC'82, 1982, pp. 267–281.
[49] J. Leroux, The general vector addition system reachability problem by Presburger inductive invari-

ants, in: LiCS'09, IEEE, 2009, pp. 4–13.
[50] C. Reutenauer, The mathematics of Petri nets, Masson and Prentice, 1990.
[51] R. Lipton, The reachability problem requires exponential space, Tech. Rep. 62, Department of Computer Science, Yale University (1976).
[52] W. Czerwinski, S. Lasota, R. Lazić, J. Leroux, F. Mazowiecki, The reachability problem for Petri nets is not elementary, in: STOC'19, 2019, pp. 24–33.
[53] C. Rackoff, The covering and boundedness problems for vector addition systems, Theoretical Computer Science 6 (2) (1978) 223–231.
[54] M. F. Atig, P. Habermehl, On Yen's path logic for Petri nets, in: RP'09, Vol. 5797 of Lecture Notes in Computer Science, Springer, 2009, pp. 51–63.
[55] M. Blockelet, S. Schmitz, Model-checking coverability graphs of vector addition systems, in: MFCS'11, Vol. 6907 of Lecture Notes in Computer Science, Springer, 2011, pp. 108–119.
[56] S. Demri, On selective unboundedness of VASS, Journal of Computer and System Sciences 79 (5) (2013) 689–713.
[57] S. Göller, C. Haase, R. Lazić, P. Totzke, A polynomial-time algorithm for reachability in branching VASS in dimension one, in: ICALP'16, Vol. 55 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 105:1–105:13.
[58] P. Habermehl, On the complexity of the linear-time mu-calculus for Petri nets, in: ICATPN'97, Vol. 1248 of Lecture Notes in Computer Science, Springer, 1997, pp. 102–116.
[59] A. Sistla, E. Clarke, The complexity of propositional linear temporal logic, Journal of the ACM 32 (3) (1985) 733–749.
[60] M. Vardi, P. Wolper, Reasoning about infinite computations, Information and Computation 115 (1994) 1–37.
[61] S. Schewe, Tighter bounds for the determinisation of Büchi automata, in: FoSSaCS'09, Vol. 5504 of Lecture Notes in Computer Science, Springer, 2009, pp. 167–181.
[62] S. Demri, P. Schnoebelen, The complexity of propositional linear temporal logics in simple cases, Information and Computation 174 (1) (2002) 84–103.
[63] T. Wilke, CTL^+ is exponentially more succinct than CTL, in: FST&TCS'99, Vol. 1999 of LNCS, Springer, 1999, pp. 110–121.
[64] K. Verma, J. Goubault-Larrecq, Karp-Miller Trees for a Branching Extension of VASS, Discrete Mathematics and Theoretical Computer Science 7 (2005) 217–230.
[65] S. Demri, M. Jurdziński, O. Lachish, R. Lazić, The covering and boundedness problems for branching vector addition systems, Journal of Computer and System Sciences 79 (1) (2013) 23–38.
[66] F. Mazowiecki, M. Pilipczuk, Reachability for Bounded Branching VASS, in: CONCUR'19, Vol. 140 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 28:1–28:13.
[67] J. Raskin, M. Samuelides, L. Van Begin, Games for counting abstractions, Electronic Notes in Theoretical Computer Science 128 (6) (2005) 69–85.
[68] N. Lynch, Log Space recognition and translation of parenthesis languages, Journal of the ACM 24 (4) (1977) 583–590.
[69] D. Della Monica, A. Murano, Parity-energy ATL for qualitative and quantitative reasoning in MAS, in: AAMAS'18, 2018, pp. 1441–1449.
[70] V. Goranko, A. Kuusisto, R. Rönnholm, Game-theoretic semantics for ATL+ with applications to model checking, Information and Computation 276.
[71] K. Chatterjee, L. Doyen, Energy parity games, Theoretical Computer Science 458 (2012) 49–60.
[72] V. Malvone, A. Murano, L. Sorrentino, Concurrent multi-player parity games, in: AAMAS'16, ACM, 2016, pp. 689–697.