Parity-energy ATL for qualitative and quantitative reasoning in MAS

Paper #XXX

ABSTRACT

In this paper, we introduce a new logic suitable to reason about strategic abilities of multi-agent systems where (teams of) agents are subject to qualitative (parity) and quantitative (energy) constraints and where goals are represented, as usual, by means of temporal properties. We formally define such a logic, named parity-energy-ATL (pe-ATL, for short), and we study its model checking problem, which we prove to be decidable with different complexity upper bounds, depending on different choices for the energy range.

1 INTRODUCTION

In recent years, game theory has been demonstrated to be very useful in open-system verification, where the game evolution emerges from the coordination of different parts viewed as autonomous and proactive agents [10, 17]. This has encouraged the development of several frameworks aimed at reasoning about strategies and their interaction [2, 13, 15, 16, 19].

An important contribution in this field has been the development of *Alternating-Time Temporal Logic* (ATL, for short) by Alur, Henzinger, and Kupferman [2]. Formally, it is obtained as a generalization of the branching-time logic CTL [9], where the path quantifiers *there exists* "E" and *for all* "A" are replaced with strategic modalities of the form "$\langle\langle A \rangle\rangle$" and "$[[A]]$", for a set A of *agents*. These modalities are used to express cooperation and competition among agents in order to achieve a temporal goal. Several decision problems have been investigated about ATL. In particular, the model checking problem is proved to be solvable in polynomial time [2].

ATL can efficiently express qualitative objectives, that is, specific temporal properties that are required to hold by coalitions of agents. Conversely, ATL (at least in the classical definition) cannot be used to express quantitative objectives. This is unfortunate as games dealing with quantitative aspects are recently receiving a lot of attention in the context of automated design and synthesis [21]. Even more, quantitative games are central in economics, where players aim at optimizing a desired payoff.

In the context of games equipped with quantitative objectives, an important contribution is given by *energy parity games* [7]. These are games played on weighted graphs in which (i) states are partitioned between two players, namely player 0 and player 1, (ii) a priority is associated to each state, and (iii) an integer weight is associated to each edge. In each round, the player owning the state chooses an outgoing edge to a successor state. The play consisting of an infinite number of rounds is won by player 0 if (i) the least priority occurring infinitely often is even and (2) the sum of the weights (namely the *energy*) along the play remains always positive.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), July 2018, Stockholm, Sweden

https://doi.org/doi

Deciding an energy parity game amounts to checking if there is an initial energy level such that player 0 has a strategy to maintain the level of energy positive while satisfying the parity condition. This problem is know to lie in NP $\cap$ coNP, as it is for parity games.

In this paper we combine energy parity games and ATL specifications in a new logical formalisms, named *parity-energy-ATL* (pe-ATL), and we solve the related model checking question. Roughly speaking, pe-ATL allows one to check the satisfaction of a parity condition while keeping the energy level within a given range along system evolutions determined by coalitions along the ATL formula. We show that the addressed model checking question lies in P, NP, or EXPTIME, depending on the type of energy range given in input.

The conceived framework can be successfully used in several contexts. In practice, it can be used in smart-city applications, which are multi-agent systems where one has to deal with both energy constraints and temporal goals [6, 20]. As another application scenario, consider a multi-agent system model for task allocation in which every odd priority represents a task request, whose allocation is represented by an even smaller priority (similar to what happens in *prompt parity games* [12, 18]). Assume now that along the process we need to use, as well as recharge, some working energy, represented by the weights over the edges. Finally, assume that we want to check that no matter which tasks are supplied by a set of *Request* agents, the *Allocation* agents can always guarantee a correct allocation. Then, such a scenario can be easily modeled and verified using our setting. Notice that the fairness requirements such as the above one can be expressed by ATL* formulas, at the price of a more expensive model checking procedure (2ExpTime-complete).

Related works. The interest in combining qualitative and quantitative reasoning in multi-agent systems has grown in the last decade. In [5], a logical framework combining agents abilities to achieve quantitative and qualitative objectives is proposed. In [3, 8], the authors propose the use of preference models that combine qualitative and quantitative objectives. More recently, in [14] a well-behaved model combining qualitative (Büchi) and quantitative (energy) conditions in a lexicographic setting is proposed.

However, the setting we investigate is new. The logic proposed in [5] flows quickly into undecidability. The other aforementioned works adopt a game-theoretic perspective, mostly focusing on the search for equilibria subject to lexicographic preferences rather than aiming at developing a logical framework for reasoning about these games. On the other hand, some proposal towards enriching ATL with resource constraints has been presented in [1, 4, 11], but none of the logics presented there deals with parity objectives.

Outline. In Section 2, we introduce the logic pe-ATL and the model checking problem for it. Sections 3 and 4 contain our technical contribution: they address, respectively, easy and difficult issues towards solving the pe-ATL model checking problem. In Section 5, we put our results together to provide a solution and a complexity analysis for the pe-ATL model checking problem, and we outline future research directions.

2 pe-ATL: QUALITATIVE AND QUANTITATIVE REASONING IN MAS

We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{N}^{>0}$ the set of positive naturals, i.e., $\mathbb{N}^{>0} = \mathbb{N}\setminus\{0\}$, and by S^n the set of vectors of n components ranging over S, for every set S and $n \in \mathbb{N}^{>0}$. According to the standard notation, for any given set S we denote by S^* (resp., S^ω) the set of finite (resp., infinite) words over the alphabet S. The length of a word $\rho \in S^* \cup S^\omega$ is denoted by $|\rho|$ (we assume $|\rho| = +\infty$ for $\rho \in S^\omega$). Finally, for every $\rho \in S^* \cup S^\omega$ and $i, j \in \mathbb{N}^{>0}$, with $i \le j \le |\rho|$, we write $\rho[i]$ to refer to the i^{th} element of ρ and $\rho[i, j]$ to refer to the finite sequence $\rho[i]\rho[i+1]\ldots\rho[j-1]\rho[j] \in S^*$.

2.1 Concurrent game structures and strategies

We fix a finite, non-empty set $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_n\}$ of n agents and a finite set $\mathcal{A}$ of atomic propositions. A subset of $\mathbb{A}\mathrm{gt}$ is called *team*.

Definition 2.1 (CGS). A *concurrent game structure* (*CGS*) G (over $\mathbb{A}\mathrm{gt}$ and $\mathcal{A}$) is a tuple $\langle Q, q^{init}, \pi, d, \delta\rangle$, where:
- Q is the finite set of *states*;
- $q^{init} \in Q$ is a distinguished state in Q, called *initial state*;
- $\pi : \mathcal{A} \to 2^Q$ is the *evaluation function*, which determines the states where propositions hold true;
- $d : Q \times \mathbb{A}\mathrm{gt} \to \mathbb{N}^{>0}$ is the *action function*: $d(q, a)$ denotes the (number of) *actions* available to an agent $a \in \mathbb{A}\mathrm{gt}$ at a state $q \in Q$. For each $q \in Q$ we denote by $D(q)$ the set $\{1, \ldots, d(q, a_1)\} \times \ldots \times \{1, \ldots, d(q, a_n)\}$ of *action profiles* available at q. A generic action (resp., action profile) is usually denoted by α (resp., $\vec{\alpha}$);
- $\delta : Q \times \mathbb{N}^n \to Q$ is the *transition function*, defined on pairs $(q, \vec{\alpha}) \in Q \times \mathbb{N}^n$ with $\vec{\alpha} \in D(q)$ and undefined when $\vec{\alpha} \notin D(q)$. If $\delta(q, \vec{\alpha}) = q'$ for some $q, q' \in Q$ and $\vec{\alpha} \in D(q)$, then we say that there is a *transition* from q to q' triggered by the action profile $\vec{\alpha}$ (denoted by $q \xrightarrow{\vec{\alpha}} q'$).

For every CGS G, we denote by Q_G, q^{init}_G, π_G, d_G, and δ_G its components (we omit the subscript when clear from the context).

Let G be a CGS. We generalize the notion of action profile to any given team: an *A-action profile*, with $A \subseteq \mathbb{A}\mathrm{gt}$, is a function from A to $\mathbb{N}^{>0}$. We denote by Λ_A the set of A-action profiles, and we usually use $\vec{\alpha}_A$ to denote a generic element of Λ_A. Moreover, for every $q \in Q$ and $A \subseteq \mathbb{A}\mathrm{gt}$, we denote by $D_A(q)$ the set of A-action profiles available at q, that is, $D_A(q) = \{\vec{\alpha}_A \in \Lambda_A \mid \vec{\alpha}_A(a) \le d(q, a) \text{ for every } a \in A\}$.

A *computation* (over a CGS G) is an infinite sequence of pairs $(q_1, \vec{\alpha}_1)(q_2, \vec{\alpha}_2)\ldots$ such that $q_i \xrightarrow{\vec{\alpha}_i} q_{i+1}$ for every i. If, in addition, $q_1 = q^{init}$, then the computation is *initial*. We denote by $\mathbb{C}_G$ the set of all computations over G (once again, we omit the subscript when there is no ambiguity). The *Q-projection* of a computation $c = (q_1, \vec{\alpha}_1)(q_2, \vec{\alpha}_2)\ldots \in \mathbb{C}$, denoted by $c_{|Q}$, is the infinite sequence of states $q_1 q_2 \ldots$. A *history* h is a pair (c, q), where $c = (q_1, \vec{\alpha}_1)(q_2, \vec{\alpha}_2)\ldots(q_k, \vec{\alpha}_k)$ is a (possibly empty) prefix of a computation and $q \in Q$ is such that $q_k \xrightarrow{\vec{\alpha}_k} q$, unless c is the empty sequence; we denote by $\mathbb{H}$ the set of histories.

Definition 2.2 ((memoryless) strategy). A *strategy* (for a team A over a CGS G) is a function $F_A : \mathbb{H} \to \Lambda_A$ such that $F_A(c, q) \in D_A(q)$, for every $(c, q) \in \mathbb{H}$. A *memoryless strategy* is a strategy F_A such that $F_A(c, q) = F_A(c', q)$ for every c, c', and q.

Let $\vec{\alpha}_A$ be an A-action profile, i.e., $\vec{\alpha}_A \in \Lambda_A$. We define the set of its *extensions* as $ext(\vec{\alpha}_A) = \{\vec{\alpha} \in \Lambda_{\mathbb{A}\mathrm{gt}} \mid \vec{\alpha}_A(a) = \vec{\alpha}(a) \text{ for every } a \in A\}$, and, for every $q \in Q$, we let $\delta_{\vec{\alpha}_A}(q) = \{q' \in Q \mid q' = \delta(q, \vec{\alpha}) \text{ for some } \vec{\alpha} \in ext(\vec{\alpha}) \cap D(q)\}$.

Definition 2.3 (outcome). The *outcome* of a strategy F_A from a state $q \in Q$ is the set $\mathrm{out}(F_A, q) = \{(q_1, \vec{\alpha}_1)(q_2, \vec{\alpha}_2)\ldots \in \mathbb{C} \mid q_1 = q$ and $\vec{\alpha}_i \in ext(F_A((q_1, \vec{\alpha}_1)\ldots(q_{i-1}, \vec{\alpha}_{i-1}), q_i)) \cap D(q_i)$ for all $i > 0\}$.

Let $i, g \subseteq Q$ (elements of i are called *invariants* while elements of g are called *goals*). A strategy F_A *guarantees* i from $q \in Q$ to mean that, for every $c \in \mathrm{out}(F_A, q)$, $c_{|Q}$ only features invariants ($q' \in i$, for every q' occurring in $c_{|Q}$ before the first occurrence of a state in g); F_A *guarantees* i *until* g from $q \in Q$ to mean that, for every $c \in \mathrm{out}(F_A, q)$, $c_{|Q}$ features at least one goal (state in g) and is such that only invariants occur in $c_{|Q}$ before the first occurrence of a goal ($q' \in i$, for every q' occurring in $c_{|Q}$ before the first occurrence of a state in g); finally, we say that F_A is (i, g, o)-*friendly* (with $o \in \{\mathcal{U}, \square\}$) from $q \in Q$ if:
- $o = \square$ and F_A guarantees i, or
- $o = \mathcal{U}$ and F_A guarantees i until g.

Without loss of generality, from now on we assume $i \cap g = \emptyset$.

In what follows, we refine the notions of CGS and corresponding strategies to comply with qualitative (parity condition) and quantitative (energy condition) requirements.

Definition 2.4 (parity condition). A *parity condition* (over a CGS G) is a function $p : Q \to \mathbb{N}$ assigning natural numbers to states in G.

Energy conditions are based on weight assignments. A *weight assignment* (over a CGS G) is a function $w : Q \times \Lambda_{\mathbb{A}\mathrm{gt}} \to \mathbb{Q}$ assigning a rational weight to every transition of G ($w(q, \vec{\alpha})$ is undefined whenever $\vec{\alpha} \notin D(q)$). For every $x, y \in \mathbb{Q} \cup \{-\infty, +\infty\}$, we denote by $[x, y]$ the set $\{z \in \mathbb{R} \mid x \le z \le y\}$.

Definition 2.5 (energy condition). An *energy condition* (over a CGS G) is a triple $e = \langle w, \mathcal{E}^{init}, [a, b]\rangle$, where w is a weight assignment over G, $\mathcal{E}^{init} \in [a, b]$ is the *initial energy level* of e, and $[a, b]$ is its *energy bound*, with $a \in \mathbb{Q} \cup \{-\infty\}$, $b \in \mathbb{Q} \cup \{+\infty\}$, and $a \le b$.

Definition 2.6 (pe-CGS). A *parity-energy CGS* (*pe-CGS*) is a triple $\mathcal{G} = \langle G, p, e\rangle$, where G is a CGS, and p and $e = \langle w, \mathcal{E}^{init}, [a, b]\rangle$ are, respectively, a parity and an energy condition over it. A *position* of $\mathcal{G}$ is a pair $(q, \mathcal{E}) \in Q \times \mathbb{Q}$; $(q^{init}, \mathcal{E}^{init})$ is the *initial position* of $\mathcal{G}$.

We lift any given parity condition p from the domain Q to the domain Q^ω in the natural way, by defining $\hat{p} : Q^\omega \to \mathbb{N}^\omega$ as $\hat{p}(q_1 q_2 \ldots) = p(q_1)p(q_2)\ldots \in \mathbb{N}^\omega$ for every $q_1 q_2 \ldots \in Q^\omega$. Moreover, for $\rho \in Q^\omega$, we let $\mathsf{infinite}_p(\rho)$ be the set of naturals that occur infinitely many times in $\hat{p}(\rho)$. The *parity* of an infinite word $\rho \in Q^\omega$ wrt. p is defined as $\min(\mathsf{infinite}_p(\rho))$.

Let $e = \langle w, \mathcal{E}^{init}, [a, b]\rangle$ be an energy condition. For every computation $c = (q_1, \vec{\alpha}_1)(q_2, \vec{\alpha}_2)\ldots$, the *energy-contribution* of a prefix $c[1, k]$ of c ($k \in \mathbb{N}^{>0}$) wrt. e, denoted by $\text{e-contrib}(e, c[1, k])$, is $\mathcal{E}^{init} + \sum_{i=1}^{k} w(q_i, \vec{\alpha}_i)$; the *energy bottom* of c wrt. e, denoted by $\mathsf{e}(e, c)$, is defined as $\mathsf{e}(e, c) = \inf_{k \to \infty} \text{e-contrib}(e, c[1, k])$; the *energy peak* of c wrt. e is defined analogously, that is, $\mathsf{E}(e, c) = \sup_{k \to \infty} \text{e-contrib}(e, c[1, k])$. Finally, for any given strategy F_A and $q \in Q$, the *energy range* of F_A from q wrt. e, denoted by $\text{e-range}(e, F_A, q)$, is the set $[\min, \max]$, where $\min = \inf\{\mathsf{e}(e, c') \mid c' \in \mathrm{out}(F_A, q)\}$ and $\max = \sup\{\mathsf{E}(e, c') \mid c' \in \mathrm{out}(F_A, q)\}$.

We are now ready to define strategies that are compliant with parity- and/or energy-conditions.

Definition 2.7 ((p,e)-strategy). Let $\mathcal{G} = \langle G, p, e\rangle$ be a pe-CGS, with $e = \langle w, \mathcal{E}^{init}, [a,b]\rangle$ and $q \in Q$. A strategy F_A is said to be:
- *p-compliant* (or a *p-strategy*) from q if for every $c \in \text{out}(F_A, q)$ $c_{|Q}$ has even parity wrt. p;
- *e-compliant* (or a *e-strategy*) from q if its energy range from q wrt. e is within the energy bound of e, i.e., $\text{e-range}(e, F_A, q) \subseteq [a,b]$;
- (p,e)*-compliant* (or a (p,e)*-strategy*) from q if it is both p- and e-compliant from q.

Let $\mathcal{G} = \langle G, p, e\rangle$ be a pe-CGS with $e = \langle w, \mathcal{E}^{init}, [a,b]\rangle$. Before concluding this sub-section, we observe that a history (c, q) univocally identifies in $\mathcal{G}$ the position $(q, \mathcal{E})$, where $\mathcal{E} = \mathcal{E}^{init} + \text{e-contrib}(e, c)$; therefore, we say that history (c, q) *leads* (in $\mathcal{G}$) to position $(q, \mathcal{E})$.

2.2 Syntax and semantics

The syntax of pe-ATL is the same as the one for ATL [2], and it is given by the following grammar:

$$\varphi ::= \mathfrak{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A\rangle\!\rangle \bigcirc \varphi \mid \langle\!\langle A\rangle\!\rangle \varphi \mathcal{U} \varphi \mid \langle\!\langle A\rangle\!\rangle \Box\varphi,$$

where $\mathfrak{p} \in \mathcal{A}$ and $A \subseteq \mathbb{A}\text{gt}$. Constants $\top$ and $\bot$, as well as other Boolean connectives and team operators (e.g., $\vee$ and $\langle\!\langle A\rangle\!\rangle\Diamond$), can be seen as abbreviations (e.g., $\langle\!\langle A\rangle\!\rangle\Diamond\varphi$ is a shorthand for $\langle\!\langle A\rangle\!\rangle\top\mathcal{U}\varphi$).

Formulas of pe-ATL are interpreted wrt. (states of) pe-CGS's. Let $\mathcal{G} = \langle G, p, e\rangle$ be a pe-CGS and $q \in Q$. The truth of a pe-ATL formula over $\mathcal{G}$ and q is inductively defined by the following clauses:
- $\mathcal{G}, q \models \mathfrak{p}$ iff $q \in \pi(\mathfrak{p})$;
- $\mathcal{G}, q \models \neg\varphi$ iff it is not the case that $\mathcal{G}, q \models \varphi$;
- $\mathcal{G}, q \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{G}, q \models \varphi_1$ and $\mathcal{G}, q \models \varphi_2$;
- $\mathcal{G}, q \models \langle\!\langle A\rangle\!\rangle \bigcirc \varphi$ iff there exists a (p,e)-strategy F_A from q such that $\mathcal{G}, c_{|Q}[2] \models \varphi$ for every $c \in \text{out}(F_A, q)$;
- $\mathcal{G}, q \models \langle\!\langle A\rangle\!\rangle\varphi_1\mathcal{U}\varphi_2$ iff there exists a (p,e)-strategy F_A from q such that for every $c \in \text{out}(F_A, q)$ there is $i \in \mathbb{N}^{>0}$ for which $\mathcal{G}, c_{|Q}[i] \models \varphi_2$ and for every $j \in \mathbb{N}$ with $1 \le j < i$ it holds $\mathcal{G}, c_{|Q}[j] \models \varphi_1$;
- $\mathcal{G}, q \models \langle\!\langle A\rangle\!\rangle\Box\varphi$ iff there exists a (p,e)-strategy F_A from q such that for every $c \in \text{out}(F_A, q)$ and every $i \in \mathbb{N}^{>0}$ it holds $\mathcal{G}, c_{|Q}[i] \models \varphi$.

Using standard notation, given a pe-ATL formula φ and a pe-CGS $\mathcal{G}$, we write $[\![\varphi]\!]_{\mathcal{G}}$ to denote the set $\{q \mid \mathcal{G}, q \models \varphi\}$, and we omit the subscript when there is no risk of ambiguity, thus writing, e.g., $[\![\varphi]\!]$ instead of $[\![\varphi]\!]_{\mathcal{G}}$. Therefore, the clauses for the operators $\mathcal{U}$ and $\Box$ can be rewritten as follows:
- $\mathcal{G}, q \models \langle\!\langle A\rangle\!\rangle\varphi_1\mathcal{U}\varphi_2$ iff there exists a $([\![\varphi_1]\!] \setminus [\![\varphi_2]\!], [\![\varphi_2]\!], \mathcal{U})$-friendly (p,e)-strategy F_A from q. $(*)$
- $\mathcal{G}, q \models \langle\!\langle A\rangle\!\rangle\Box\varphi_1$ iff there exists a $([\![\varphi_1]\!], \emptyset, \Box)$-friendly (p,e)-strategy F_A from q. $(\star)$

By simultaneously replacing $\models_{\text{p}}$ for $\models$ and p-strategy for (p,e)-strategy, or, alternatively, $\models_{\text{e}}$ for $\models$ and e-strategy for (p,e)-strategy, we obtain two alternative semantic definitions, namely the p- and the e-semantics. We name the resulting logics p-ATL and e-ATL, respectively.

Given a pe-ATL formula φ and a pe-CGS $\mathcal{G}$, we say that $\mathcal{G}$ satisfies φ, denoted by $\mathcal{G} \models \varphi$, if $\mathcal{G}, q^{init} \models \varphi$. Moreover, we say that $\mathcal{G}$ satisfies φ in the p-semantics (resp., e-semantics), denoted by $\mathcal{G} \models_{\text{p}} \varphi$ (resp., $\mathcal{G} \models_{\text{e}} \varphi$), if, and only if, $\mathcal{G}, q^{init} \models_{\text{p}} \varphi$ (resp., $\mathcal{G}, q^{init} \models_{\text{e}} \varphi$).

2.3 The model checking problem

The *model checking* problem for pe-ATL (resp., p-ATL, e-ATL) consists in verifying, given a pe-CGS $\mathcal{G}$ and a pe-ATL formula φ, whether $\mathcal{G} \models \varphi$ (resp., $\mathcal{G} \models_{\text{p}} \varphi$, $\mathcal{G} \models_{\text{e}} \varphi$) holds. It is easy to show that the model checking problem for p-ATL (resp., e-ATL) can be reduced to the model checking problem for pe-ATL. Indeed, for a $\mathcal{G} = \langle G, p, e\rangle$, with $e = \langle w, \mathcal{E}^{init}, [a,b]\rangle$, let *pcgs* and *ecgs* be defined as:
- $pcgs(\mathcal{G}) = \langle G, p, e'\rangle$, where $e' = \langle w', \mathcal{E}, [a,b]\rangle$, $\mathcal{E} = a$, and $w'(q, \vec{\alpha}) = 0$ for every $q \in Q$ and every $\vec{\alpha} \in D(q)$;
- $ecgs(\mathcal{G}) = \langle G, p', e\rangle$, with $p'(q) = 0$ for every $q \in Q$.

The following results easily hold.

Proposition 2.8. *For every pe-CGS $\mathcal{G}$ and pe-ATL formula φ:*
- *$\mathcal{G} \models_{\text{p}} \varphi$ if and only if $pcgs(\mathcal{G}) \models \varphi$, and*
- *$\mathcal{G} \models_{\text{e}} \varphi$ if and only if $ecgs(\mathcal{G}) \models \varphi$.*

Lemma 2.9. *There are polynomial time reductions from the model checking problem for p-ATL to the one for pe-ATL and from the model checking problem for e-ATL to the one for pe-ATL.*

Without loss of generality, we only consider *integer energy conditions*, i.e., energy conditions $e = \langle w, \mathcal{E}^{init}, [a,b]\rangle$ such that w ranges over integer, $\mathcal{E}^{init} \in \mathbb{Z}$, and $a, b \in \mathbb{Z} \cup \{-\infty, +\infty\}$. Details on how to convert a general model checking instance into an equivalent one featuring an integer energy condition are omitted for lack of space. We fix a pe-CGS $\mathcal{G} = \langle G, p, e\rangle$, where $e = \langle w, \mathcal{E}^{init}, [a,b]\rangle$ is an integer energy condition. Moreover, we ignore the case where $a = -\infty$ and $b \neq +\infty$, as it can be dealt with analogously to the one where $a \neq -\infty$ and $b = +\infty$, and we only consider the remaining three cases: (*i*) $[a,b] = [-\infty, +\infty]$ (*unbounded instances*), (*ii*) $a, b \in \mathbb{Z}$ (*bounded instances*), (*iii*) $a \in \mathbb{Z}$ and $b = +\infty$ (*mixed instances*).

3 SOLVING THE EASY CASES

The hardest case to solve when addressing the model checking problem for pe-ATL concerns formulas of the kind $\langle\!\langle A\rangle\!\rangle\varphi_1\mathcal{U}\varphi_2$. As a consequence of $(*)$ and $(\star)$, deciding these formulas amounts to establishing whether there exists an (i, g, o)-friendly (p,e)-strategy F_A, for a suitable $o \in \{\mathcal{U}, \Box\}$, denoting the operator, and suitable sets i and g denoting the semantics of φ_1 and φ_2, respectively.

In this section, we address this latter problem. More precisely, we devise a procedure to decide if there is an (i, g, o)-friendly (p,e')-strategy for a team A from a state q, for any given i, g, o, A, and q, and where $e' = \langle w, \mathcal{E}, [a,b]\rangle$ for a given $\mathcal{E}$, that is, e' is obtained from $e = \langle w, \mathcal{E}^{init}, [a,b]\rangle$ by replacing its initial energy level $\mathcal{E}^{init}$ with the given one $\mathcal{E}$.

We focus here on bounded and unbounded instances, that is, those in which either $[a,b] = [-\infty, +\infty]$ or $a \neq -\infty$ and $b \neq +\infty$. These cases are easier to deal with. In the next section, we will handle the more complex mixed instances.

Unbounded instances can be treated as particular cases of bounded ones. Indeed, an unbounded instance having energy condition $e = \langle w, \mathcal{E}^{init}, [-\infty, +\infty]\rangle$ can be easily converted into an equivalent bounded one by replacing e with $e' = \langle w', 0, [0,0]\rangle$, where w' assign weight 0 to every transition. In what follows, we focus on bounded instances, unless differently specified.

The algorithm hinges on some characterizations of strategies (Lemma 3.2 and Corollary 3.3 below) in terms of memoryless and uniform ones, the latter being defined as follows.

Definition 3.1 (uniform strategies). A *uniform strategy* is a strategy F_A such that $F_A(c, q) = F_A(c', q)$ for every c, c', and q such that $\text{e-contrib}(e, c) = \text{e-contrib}(e, c')$.

Lemma 3.2. *If $a \neq -\infty$ and $b \neq +\infty$ (bounded instance), then*
(*a*) *a (p,e)-strategy from q exists if and only if a uniform one exists,*
(*b*) *for every $i, g \in Q$, an $(i, g, \mathcal{U})$-friendly strategy from q exists if and only if a uniform one exists, and*
(*c*) *for every $i \in Q$, an $(i, \emptyset, \square)$-friendly (p,e)-strategy from q exists if and only if a uniform one exists.*

Notice also that, despite the results stated in Lemma 3.2 and unlike the case of $(i, g, \square)$-friendly (p,e)-strategies, an $(i, g, \mathcal{U})$-friendly (p,e)-strategy is not necessarily uniform, as the strategy can associate different action profiles to the same position depending on which is the current phase: intuitively, a strategy is followed to satisfy the $(i, , \mathcal{U})$-friendliness and another one is adopted to meet the parity condition.

Clearly, a memoryless strategy (Definition 2.2) is also uniform. Even if the converse is not necessarily true, both kinds of strategies can be considered to be *positional*: two histories (c, q), (c', q) with e-contrib(e, c) = e-contrib(e, c') lead to the same position $(q, \mathcal{E})$ of $\mathcal{G}$, and thus a uniform strategy F_A forces A to behave uniformly whenever the game is in the same position. It is also worth noticing that, as far as bounded instances are concerned, a uniform strategy is a *bounded-memory* one, i.e., the strategy proposes the same action profile for pairs of histories (c, q) and (c', q) such that the suffixes of c and c' coincide up to a certain constant bound: indeed, since the size of the set of positions is $\lambda = |Q| \times (b - a + 1)$, a uniform strategy is a bounded-memory one with bound λ (λ-memory strategy). Memoryless strategies can be thought of as 0-memory ones.

As we have already observed, unbounded instances can be treated as special bounded ones where energy never changes. Thus, there is a bijection between states and reachable positions, from which the following corollary follows.

Corollary 3.3. *If $[a, b] = [-\infty, +\infty]$ (unbounded instance), then*
- *a (p,e)-strategy from q exists if and only if a memoryless one exists,*
- *for every $i, g \in Q$, an $(i, g, \mathcal{U})$-friendly strategy from q exists if and only if a uniform one exists, and*
- *for every $i \in Q$, an $(i, \emptyset, \square)$-friendly (p,e)-strategy from q exists if and only if a uniform one exists.*

Having these results in mind, it is not difficult to devise algorithms to look for (i, g, o)-friendly (p,e)-strategies (see Algorithm 1 for full details). In the following, we give an intuitive description only for the case when $o = \mathcal{U}$, i.e., $(i, g, \mathcal{U})$-friendly (p,e)-strategies.

The algorithm consists of two phases. During the first phase, it focuses on the $(i, g, \mathcal{U})$-friendliness: it explores all strategies for A trying to reach a goal in g while guaranteeing invariants in i. Once a goal is reached, the second phase begins, during which the algorithm focuses on the parity condition by searching for the existence of a (p,e)-strategy. The step from the first phase to the second one is triggered when a goal is reached, at which point the flag variable igU_friendly is set to TRUE and the history is reset (lines 6–8 in Algorithm 1). Notice that when searching for $(i, g, \square)$-friendly (p,e)-strategies, no goal is ever reached (g is in fact set to empty at line 2 in Algorithm 1), so the history is never reset and the search is performed in only one phase; this is coherent with the result in Lemma 3.2 (*c*), which states that a uniform strategy is enough to search for $(i, g, \square)$-friendly (p,e)-strategies. Throughout the whole process, the energy condition is checked as well, that is, the algorithm verifies that in each position $(q, \mathcal{E})$ the energy level $\mathcal{E}$ is within the allowed range $[a, b]$.

Alg. 1 For bounded instances, it checks for the existence of a (i, g, o)-friendly (p, e')-strategy for A from q, where $e' = \langle w, \mathcal{E}, [a, b] \rangle$.

```
1: procedure ∃-STRATEGY-BOUNDED(𝒢, i, g, o, A, q, ℰ, history, igU_friendly)
                  ▹ 𝒢 = ⟨G, p, e⟩, G = ⟨Q, q^init, π, d, δ⟩, e = ⟨w, ℰ^init, [a, b]⟩
2:     if o = □ then g ← ∅      ▹ no goals when searching for (i, g, o)-friendly (p, e′)-strategy
3:     if ℰ ∉ [a, b] then return FALSE
4:     if not igU_friendly then
5:         if q ∉ i ∪ g then return FALSE
6:         if q ∈ g then                                  ▹ when a goal is reached …
7:             igU_friendly ← TRUE                        ▹ … a flag is set to true,
8:             DELETE-HISTORY(history)                    ▹ … history is reset,
                                                          ▹ … and the second phase begins
9:     if history[q, ℰ] ≠ NULL then         ▹ position (q, ℰ) is visited for the second time
10:        if o = 𝒰 and igU_friendly and history[q, ℰ] is even then return TRUE
11:        if o = □ and history[q, ℰ] is even then return TRUE
12:        return FALSE
13:    else
14:        UPDATE-HISTORY(history, a, b, p, q, ℰ)
15:        found_strategy ← FALSE
16:        for α⃗_A ∈ D_A(q) do                          ▹ cycle over proponent's strategies
17:            good_strategy ← TRUE
18:            for α⃗ ∈ D(q) ∩ ext(α⃗_A) do               ▹ cycle over opponent's strategies
19:                q ← δ(q, α⃗)
20:                ℰ ← ℰ + w(q, α⃗)
21:                if not ∃-STRATEGY-BOUNDED(𝒢, i, g, o, A, q, ℰ, history, igU_friendly)
22:                      then good_strategy ← FALSE
23:            if good_strategy then found_strategy ← TRUE
24:        if found_strategy then return TRUE
25:        else return FALSE

26: procedure UPDATE-HISTORY(history, a, b, p, q, ℰ)
        ▹ history keeps track of all visited positions and, for each such positions, stores the min-
          imum parity occurred since the last visit to that position
27:    history[q, ℰ] ← p(q)
28:    for (q′, ℰ′) ∈ Q × [a, b] s.t. history[q, ℰ] ≠ NULL do
29:        history[q′, ℰ′] ← min{history[q′, ℰ′], p(q)}
```

In both phases, a strategy is discharged if during one of its outcomes the same position is reached twice without reaching a goal (in the first phase) or being able to guarantee the parity condition by, intuitively, reaching a cycle with even parity (in the second phase). Observe that it is safe to adopt such a termination condition thanks to Lemma 3.2.

Complexity. The computational complexity of the procedure is given from the depth of the recursion tree. In the worst case, the algorithm visit each position once in each phase (that is, before and after reaching a goal), and a recursive call is made for each such visits, thus yielding $2 \times |Q| \times (b - a + 1)$ calls. Assuming that a and b are represented in binary, the complexity of the algorithm is exponential in the size of the input.

Theorem 3.4. *The procedure* $\exists$-STRATEGY-BOUNDED *runs in time exponential in the size of the input.*

The following corollary comes in handy to deal with unbounded instances, as they can be treated as bounded ones where $a = b = 0$.

Corollary 3.5. *If $a = b$, then the procedure* $\exists$-STRATEGY-BOUNDED *runs in time polynomial in the size of the input.*

4 SOLVING THE DIFFICULT MIXED CASE

In this section we deal with mixed instances, where the energy bound is bounded below and unbounded above. This case is much technical involved (the case in which the energy bound is bounded only above can be dealt with analogously and thus omitted).

We first observe in Section 4.1 that there is a natural correspondence between strategies (resp., (p,e)-strategies) for a team A and a suitable class of infinite trees, named *A-trees* (resp., *(p,e)-A-trees*), the latter being easier to manipulate and deal with.

Then, in Section 4.2 we define appropriate finite structures, named *witnesses*, which are shown, in Sections 4.3 and 4.4, to be expressively complete for (p,e)-strategies, meaning that every such witness corresponds to a particular (p,e)-strategy (Section 4.3), and, vice versa, every (p,e)-strategy can be compactly encoded into a witness which keeps enough information about the strategy itself (Section 4.4). As a consequence, the search for a (p,e)-strategy amounts to looking for a witness for it.

Finally, we establish a bound for the size of a witness corresponding to a strategy; thus, the search space to search for witnesses is finite, and a decision procedure follows.

4.1 Tree-based representation for strategies

Trees are particularly apt to express the possible evolutions of a multi-agent system (represented as a pe-CGS) that are consistent with a strategy adopted by a team of agents.

A *node* is a tuple $N = \langle q, \mathcal{E}, \vec{\alpha}, N_1 \dots N_k \rangle$, where $q \in Q$, $\mathcal{E} \in \mathbb{Z}$ is the *energy level* associated with the node, $\vec{\alpha} \in D(q) \cup \{\#\}$ (# is a placeholder for an undefined action profile; it is used for root nodes), and $N_1 \dots N_k$ is a finite (possibly empty) sequence of nodes, representing the path to N. We use $\mathscr{N}$ to refer to the set of all nodes. We denote by $state(N)$, $e\text{-}level(N)$, $in\text{-}action(N)$, and $ancestors(N)$ the first, second, third, and fourth component of N, respectively, and we write $path\text{-}to(N)$ to denote the sequence obtained by enqueuing N to the sequence $ancestors(N)$, i.e., $path\text{-}to(N) = N_1 \dots N_k N$. If N_1 and N_2 are two nodes such that $N_1 \in ancestors(N_2)$, then we say that N_1 (resp., N_2) is an *ancestor* (resp., a *descendant*) of N_2 (resp., N_1); we denote by $desc(N)$ the set of descendant of a node N, i.e., $desc(N) = \{N' \in \mathscr{N} \mid N \in ancestors(N')\}$, and, for $X \subseteq \mathscr{N}$, we let $desc(X) = \bigcup_{N \in X} desc(N)$. Moreover, N_2 is a *child* of N_1 if it is an immediate descendant of N_1 (i.e., $path\text{-}to(N_1) = ancestors(N_2)$): in this case we also say that N_1 is the *father* of N_2, and we use $father(N_2)$ to denote N_1. A *root node*, or simply *root*, is a node N for which $ancestors(N) = \varepsilon$ (ε denotes the empty sequence). A *tree* $\mathcal{T}$ is a set of nodes that contains exactly one root, denoted by $root_{\mathcal{T}}$, and such that for every $N \in \mathcal{T}$ and every ancestor N' of N it holds: (i) $N' \in \mathcal{T}$, and (ii) $ancestors(N')$ is a proper prefix of $ancestors(N)$. For a tree $\mathcal{T}$ and a node $N \in \mathcal{T}$, we denote by $children_{\mathcal{T}}(N)$ the set of children of N in $\mathcal{T}$, i.e., $children_{\mathcal{T}}(N) = \{N' \in \mathcal{T} \mid N'$ is a child of $N\}$. A *leaf* of $\mathcal{T}$ is a node N that has no children in $\mathcal{T}$ (i.e., $children_{\mathcal{T}}(N) = \emptyset$); we denote by $\mathrm{leaves}_{\mathcal{T}}$ the set of leaves of $\mathcal{T}$. A *branch* $\mathcal{B}$ of $\mathcal{T}$ is a maximal subset of $\mathcal{T}$ such that every two different nodes in $\mathcal{B}$ are one an ancestor of the other. By the maximality requirement, if $\mathcal{B}$ is a branch of $\mathcal{T}$ and $N \in \mathcal{B}$ is not a leaf of $\mathcal{T}$, then $|children_{\mathcal{B}}(N)| = 1$; a branch of $\mathcal{T}$ is finite if and only if it contains a leaf of $\mathcal{T}$. (Notice that a branch of a tree is a tree itself; moreover, since a branch is a linearly ordered set of nodes, we treat it as a sequence whenever we find it convenient.)

For a (finite or infinite) sequence of nodes $\mathcal{N} = N_1 N_2 \dots$, we denote by $state(\mathcal{N})$ the sequence of states $state(N_1)state(N_2)\dots$, and by $state\text{-}action(\mathcal{N})$ the sequence of pairs $(state(N_1), in\text{-}action(N_2))$ $(state(N_2), in\text{-}action(N_3))\dots$. A node N is *(i,g)-friendly* (for $i, g \subseteq Q$) if there is $N' \in path\text{-}to(N)$ such that $state(N') \in g$ and $state(N'') \in i$ for every $N'' \in ancestors(N')$. A tree $\mathcal{T}$ is *(i,g)-friendly* if every branch $\mathcal{B}$ of $\mathcal{T}$ features at least one (i,g)-friendly node; moreover, we say that $\mathcal{T}$ is *i-invariant* if $state(N) \in i$ for every $N \in \mathcal{T}$ and that $\mathcal{T}$ *ranges within* $[a,b]$ if $e\text{-}level(N) \in [a,b]$ for every $N \in \mathcal{T}$.

For k pairs of nodes $N_1, N_1', \dots, N_k, N_k' \in \mathcal{T}$, $\mathcal{T}_{[N_1 \hookleftarrow N_1', \dots, N_k \hookleftarrow N_k']}$ is the tree obtained from $\mathcal{T}$ by replacing, for every $i = 1, \dots, k$, the sub-tree rooted in N_i with the one rooted in N_i'. Towards a formal definition, we first inductively define, for $N_1, N_2 \in \mathcal{T}$, the node transformation function $\tau^{\mathcal{T}}_{[N_1 \hookleftarrow N_2]}$: for every $N \in \mathcal{T}$ that is a descendant of N_2, with $N = \langle q, \mathcal{E}, \vec{\alpha}, path\text{-}to(N_2) N_1'' \dots N_h'' \rangle$ and $h \geq 0$

$$\begin{aligned} &\tau^{\mathcal{T}}_{[N_1 \hookleftarrow N_2]}(N) = \langle q, \mathcal{E} - e\text{-}level(N_2) + e\text{-}level(N_1), \vec{\alpha}, \mathcal{N}' \rangle, \\ &\text{where } \mathcal{N}' = path\text{-}to(N_1)\tau^{\mathcal{T}}_{[N_1 \hookleftarrow N_2]}(N_1'') \dots \tau^{\mathcal{T}}_{[N_1 \hookleftarrow N_2]}(N_h''). \end{aligned} \tag{1}$$

We omit superscript and subscript from the above notation when they are clear from the context; e.g., we simply write $\tau_{[N_1 \hookleftarrow N_2]}$ or τ in place of $\tau^{\mathcal{T}}_{[N_1 \hookleftarrow N_2]}$. Notice that τ is an injection. Then, $\mathcal{T}_{[N_1 \hookleftarrow N_1', \dots, N_k \hookleftarrow N_k']}$ is defined as:

$$\begin{aligned} &\mathcal{T} \setminus \textstyle\bigcup_{i \in \{1,\dots,k\}} \{N \mid N \text{ is a descendant of } N_i \text{ in } \mathcal{T}\} \\ &\quad \cup \textstyle\bigcup_{i \in \{1,\dots,k\}} \{\tau_{[N_i \hookleftarrow N_i']}(N) \mid N \text{ is a descendant of } N_i' \text{ in } \mathcal{T}\}. \end{aligned}$$

In order to be able to use trees to capture the possible ways pe-CGS's can evolve according to team strategies, we make use of the following notions. First, for every $(q,\mathcal{E}) \in Q \times [a,b]$, we define its *$\vec{\alpha}$-successor*, for $\vec{\alpha} \in D(q)$, as $\mathrm{succ}_{\vec{\alpha}}(q,\mathcal{E}) = (q_{\vec{\alpha}}, \mathcal{E}_{\vec{\alpha}})$, where $q_{\vec{\alpha}} = \delta(q,\vec{\alpha})$ and $\mathcal{E}_{\vec{\alpha}} = \mathcal{E} + w(q,\vec{\alpha})$. Then, we define, for a team A and $\vec{\alpha}_A \in D_A(q)$, the set $\text{succ-set}_{\vec{\alpha}_A}(q,\mathcal{E}) = \{\mathrm{succ}_{\vec{\alpha}}(q,\mathcal{E}) \mid \vec{\alpha} \in ext(\vec{\alpha}_A) \cap D(q)\}$. Finally, we lift the definition of $\mathrm{succ}_{\vec{\alpha}}$ and $\text{succ-set}_{\vec{\alpha}_A}$ to the domain of nodes: for a node N we define $\mathrm{succ}_{\vec{\alpha}}(N) = \langle q_{\vec{\alpha}}, \mathcal{E}_{\vec{\alpha}}, \vec{\alpha}, path\text{-}to(N) \rangle$, where $(q_{\vec{\alpha}}, \mathcal{E}_{\vec{\alpha}}) = \mathrm{succ}_{\vec{\alpha}}(state(N), e\text{-}level(N))$ and $\text{succ-set}_{\vec{\alpha}_A}(N) = \{\mathrm{succ}_{\vec{\alpha}}(N) \mid \vec{\alpha} \in ext(\vec{\alpha}_A) \cap D(state(N))\}$.

Definition 4.1 (A-tree). Let A be a team. An *A-strategy tree* (*A-tree* for short) rooted in $q \in Q$ is a tree $\mathcal{T}$ having $\langle q, \mathcal{E}^{init}, \#, \varepsilon \rangle$ as root and such that for every $N \in \mathcal{T}$ either N is a leaf of $\mathcal{T}$ or $children_{\mathcal{T}}(N) = \text{succ-set}_{\vec{\alpha}_A}(N)$ for some $\vec{\alpha}_A \in D_A(state(N))$. An A-tree is *partial* if it contains leaves, it is *complete* otherwise.

For every A-tree $\mathcal{T}$ and node $N \in \mathcal{T} \setminus \mathrm{leaves}_{\mathcal{T}}$, we denote by $A\text{-}profile_{\mathcal{T}}(N)$ the A-action profile $\vec{\alpha}_A \in D_A(state(N))$ such that $children_{\mathcal{T}}(N) = \text{succ-set}_{\vec{\alpha}_A}(N)$. Every complete A-tree $\mathcal{T}$ identifies a strategy, denoted by $F_A^{\mathcal{T}}$, as follows: for every $\rho = state\text{-}action(ancestors(N))$ for some $N \in \mathcal{T}$, we set $F_A^{\mathcal{T}}(\rho, state(N)) = A\text{-}profile_{\mathcal{T}}(N)$, and we set $F_A^{\mathcal{T}}(\rho, state(N))(a) = 1$ for every other (ρ, N) and every $a \in A$. Therefore, a complete A-tree $\mathcal{T}$ rooted in q describes the outcome of $F_A^{\mathcal{T}}$ from q. Conversely, it is clear that, for a given $q \in Q$, a strategy F_A univocally identifies a complete A-tree rooted in q, which we denote by $\mathcal{T}^{F_A}$.

Proposition 4.2. *For a complete A-tree $\mathcal{T}$ rooted in q, we have that* $\mathrm{out}(F_A^{\mathcal{T}}, q) = \{state\text{-}action(\mathcal{B}) \mid \mathcal{B}$ *is a branch of* $\mathcal{T}\}$*, and for a strategy F_A and $q \in Q$, we have* $\mathrm{out}(F_A, q) = \{state\text{-}action(\mathcal{B}) \mid \mathcal{B}$ *is a branch of* $\mathcal{T}^{F_A}\}$.

Definition 4.3 ((p,e)-A-tree). A *(p,e)-A-tree* is a complete A-tree $\mathcal{T}$ such that $F^{\mathcal{T}}$ is a (p,e)-strategy for A.

We omit the team whenever it is clear from the context or not relevant; e.g., we write (p,e)-tree instead of (p,e)-A-tree.

Theorem 4.4. *An $(i, g, \mathcal{U})$-friendly (p,e)-strategy from q exists if and only if there is an (i,g)-friendly (p,e)-tree rooted in q; an $(i, g, \Box)$-friendly (p,e)-strategy from q exists if and only if there is an i-invariant (p,e)-tree rooted in q.*

4.2 Witnesses

We introduce here the notion of witness (based on the ones of partial witness and accumulator, see below), and we show in the next sections that it is possible to reduce the existence of strategies (trees) to the existence of witnesses of bounded sizes. In the reminder, let A denote a team.

Definition 4.5 (pw). A *partial witness* (*pw*, for short) for A is an LTS (labeled transition system) $S = (V = V' \uplus V^{fin}, T = T^{>} \uplus T^{=} \uplus T^{<}, \ell)$, where the set of *vertices* $V \subseteq Q$ (partitioned into $\{V', V^{fin}\}$), the set of *transitions* $T \subseteq \delta$ (partitioned into $\{T^{>}, T^{=}, T^{<}\}$), and the *labeling function* $\ell : V \to \mathbb{Z}$, associating an energy level with each vertex, are subjects to the following constraints:

- for every $q \in V$ there is $(q, \vec{\alpha}, q') \in T$ if and only if $q \in V'$;
- for every $q \in V$, $\ell(q) \geq a$;
- for every $\sim \in \{<, =, >\}$ and every $(q, \vec{\alpha}, q') \in T^{\sim}$, $\ell(q) + w(q, \vec{\alpha}) \sim \ell(q')$;
- for every $q \in V'$ there is a unique $\vec{\alpha}_A \in D_A(q)$ for which $D(q) \cap ext(\vec{\alpha}_A) = \{\vec{\alpha} \mid (q, \vec{\alpha}, q') \in T$ for some $q'\}$; we denote by $A\text{-}profile_S(q)$ such an unique A-action profile $\vec{\alpha}_A$.

Unless otherwise stated, we use the following notation: the first component of a pw S (resp., S_i, with $i \in \{1, 2, 3, 4\}$) is denoted by V (resp., V_i), its second component is denoted by T (resp., T_i), and its third component is denoted by ℓ (resp., ℓ_i); moreover, even if not explicitly said, V (resp., V_i) is assumed to be partitioned into $\{V', V^{fin}\}$ (resp., $\{V'_i, V_i^{fin}\}$), while T (resp., T_i) is assumed to be partitioned into $\{T^{>}, T^{=}, T^{<}\}$ (resp., $\{T_i^{>}, T_i^{=}, T_i^{<}\}$).

Let S be a pw. We denote by $V^{<}$ the set $\{q \in V' \mid (q, \vec{\alpha}, q') \in T^{<}$ for some $\vec{\alpha}$ and $q'\}$; we say that S is *(i, g)-friendly* if $V' \subseteq i$ and $V^{fin} \subseteq g$ and that S is *i-invariant* if $V \subseteq i$. Moreover, we define an *S-path* from q_1 to q_r as a sequence $\sigma = \langle q_1 \vec{\alpha}_1 q_2 \ldots q_{r-1} \vec{\alpha}_{r-1} q_r \rangle$ with $r > 1$ and $(q_i, \vec{\alpha}_i, q_{i+1}) \in T$ for every $i \in \{1, \ldots, r-1\}$; if $(q_i, \vec{\alpha}_i, q_{i+1}) \in T^{=} \cup T^{>}$ for every $i \in \{1, \ldots, r-1\}$ and $(q_i, \vec{\alpha}_i, q_{i+1}) \in T^{>}$ for at least one $i \in \{1, \ldots, r-1\}$, then σ is said to be *increasing*. We use the notation $q_1 \Rightarrow_S q_r$ to denote the existence of an S-path from q_1 to q_r and we denote by $\sigma_{|V}$ the restriction of σ to elements of V, i.e., $\sigma_{|V} = q_1 q_2 \ldots q_r$. Finally, an *$S$-cycle* is an S-path $q_1 \Rightarrow_S q_r$ with $q_1 = q_r$.

Definition 4.6 (accumulator). An *A-accumulator* is a pair $\mathcal{A} = (S_1, S_2)$ of pw's for A such that:

- $T_2^{<} = \emptyset$,
- every S_2-cycle is increasing,
- every $q \in V'_2$ occurs in some S_2-cycle,
- $V'_2 \subseteq V'_1$,
- $V_2^{fin} \subseteq V_1$ and $\ell_2(q) \geq \ell_1(q)$ for every $q \in V_2^{fin}$,
- $V_1^{<} \subseteq V'_2$ and $\ell_1(q) \geq \ell_2(q)$ for every $q \in V_1^{<}$.

Let $\mathcal{A} = (S_1, S_2)$ be an A-accumulator. It is said to be *acyclic* if it features no S_1-cycles and $\ell_2(q) > \ell_1(q)$ holds for every $q \in V_2^{fin}$ for which there is $q' \in V'_2$ such that $q' \Rightarrow_{S_2} q$ and $q \Rightarrow_{S_1} q'$. Its *parity function* wrt. p, denoted by $p^{\mathcal{A}} : V_1 \to \mathbb{N}$, is defined as:

$$p^{\mathcal{A}}(q) = \begin{cases} \min\{p(q') \mid q \Rightarrow_{S_2} q' \text{ and } q' \Rightarrow_{S_2} q\} & \text{if } q \in V_1^{<} \\ p(q) & \text{otherwise.} \end{cases}$$

$\mathcal{A}$ has *even parity* if $p^{\mathcal{A}}(\sigma) = \min\{p^{\mathcal{A}}(q) \mid q \in \sigma_{|V}\}$ is even, for every S_1-cycle σ.

Definition 4.7 (witness). A *$\mathcal{U}$-witness* for (A, i, g) is a quadruple $\mathcal{W} = (S_1, S_2, S_3, S_4)$, where

(a) (S_1, S_2) is an acyclic A-accumulator, with S_1 (i, g)-friendly pw,
(b) (S_3, S_4) is an A-accumulator with even parity and $V_3^{fin} = \emptyset$,
(c) $V_1^{fin} \subseteq V_3$ and $\ell_1(q) \geq \ell_3(q)$ for every $q \in V_1^{fin}$.

A $\Box$-*witness* for (A, i, g) is an i-invariant A-accumulator $\mathcal{W} = (S_1, S_2)$ with even parity and $V_1^{fin} = \emptyset$.

Notice that g plays no role in the definition of $\Box$-witness and could be omitted; however, we decided to keep it for the sake of a uniform notation. For a witness $\mathcal{W} = (S_1, S_2, S_3, S_4)$, we use $init(\mathcal{W})$ to refer to the set V_1 of vertices of S_1.

4.3 From witnesses to (p,e)-trees

We first define a transformation from a accumulators to (p,e)-trees. This gives us a way to convert a $\Box$-witnesses for (A, i) into an $(i, g, \Box)$-friendly (p,e)-trees. Then, based on such a transformation, we show how to build, from an $\mathcal{U}$-witness $\mathcal{W}$ for (A, i, g) with $q \in init(\mathcal{W})$, an $(i, g, \mathcal{U})$-friendly (p,e)-tree rooted in q.

4.3.1 Converting $\Box$-witnesses. Let $\mathcal{A} = (S_1, S_2)$ be an A-accumulator, $q \in V_1$, and $E \geq \ell_1(q)$. We define tree $\tau_{\mathcal{A},q,E}$ as the smallest set of nodes such that: (In what follows, for the sake of a lighter notation, for a node N we denote $state(N)$ by q_N and, for $i \in \{1, 2\}$, we let $\vec{\alpha}_N^i = A\text{-}profile_{S_i}(q_N)$; notice that $A\text{-}profile_{S_i}(q_N)$, and thus $\vec{\alpha}_N^i$, is undefined whenever $q_N \notin V'_i$.)

- $(q, E, \varepsilon, \varepsilon) \in \tau_{\mathcal{A},q,E}$;
- for every $N \in \tau_{\mathcal{A},q,E}$ with $q_N \in V'_1$:
  - if there is $(q_N, \vec{\alpha}, q') \in T_1$ such that $e\text{-}level(N) + w(q_N, \vec{\alpha}) < \ell_1(q')$, then $\tau'_{\mathcal{A},N} \subseteq \tau_{\mathcal{A},q,E}$, where $\tau'_{\mathcal{A},N}$ is the smallest set of nodes such that:
    (i) succ-set$_{\vec{\alpha}_N^2}(N) \subseteq \tau'_{\mathcal{A},N}$;
    (ii) for every $N' \in \tau'_{\mathcal{A},N}$, if $q_{N'} \in V'_2$ and either $e\text{-}level(N') \leq \ell_1(q_{N'})$ or there is $N'' \in ancestors(N')$ with $q_{N'} = q_{N''}$ and $e\text{-}level(N'') \geq e\text{-}level(N')$, then succ-set$_{\vec{\alpha}_{N'}^2}(N') \subseteq \tau'_{\mathcal{A},N}$;
  - if $e\text{-}level(N) \geq \ell_1(q_N)$, $e\text{-}level(N) + w(q_N, \vec{\alpha}) \geq \ell_1(q')$ for every $(\vec{\alpha}, q')$ such that $(q_N, \vec{\alpha}, q') \in T_1$, and succ-set$_{\vec{\alpha}_N^2}(N) \nsubseteq \tau_{\mathcal{A},q,E}$ (this last condition is needed to avoid expanding N following transitions of S_1 when N has already been expanded following transitions of S_2), then succ-set$_{\vec{\alpha}_N^1}(N) \subseteq \tau_{\mathcal{A},q,E}$.

Lemma 4.8. *For every A-accumulator $\mathcal{A} = (S_1, S_2)$, every $q \in V_1$, and every $E \geq \ell_1(q)$, $\tau_{\mathcal{A},q,E}$ is an A-tree rooted in q and ranging within $[a, +\infty]$. Moreover:*

- *$\tau_{\mathcal{A},q,E}$ is complete if and only if $V_1^{fin} = \emptyset$ and*
- *$\tau_{\mathcal{A},q,E}$ is finite if and only if $\mathcal{A}$ is acyclic.*

Lemma 4.9. *If $\mathcal{A} = (S_1, S_2)$ is an acyclic A-accumulator, with S_1 being an (i, g)-friendly pw, then $\tau_{\mathcal{A},q,\ell_1(q)}$ is a finite (i, g)-friendly A-tree ranging within $[a, +\infty]$ and rooted in q, for every $q \in V_1$.*

Lemma 4.10. *If $\mathcal{A} = (S_1, S_2)$ is an A-accumulator with even parity and $V_1^{fin} = \emptyset$, then $\tau_{\mathcal{A},q,\ell_1(q)}$ is a (p,e)-tree rooted in q, for all $q \in V_1$.*

Corollary 4.11. *For every $\Box$-witness $\mathcal{W}$ for (A, i, g) and every $q \in init(\mathcal{W})$, $\tau_{\mathcal{W},q,\ell_1(q)}$ is an i-invariant (p,e)-tree rooted in q.*

4.3.2 Converting $\mathcal{U}$-witnesses. Let $\mathcal{W} = (S_1, S_2, S_3, S_4)$ be a $\mathcal{U}$-witness for (A, i, g) and $q \in init(\mathcal{W})$. We define $\tau_{\mathcal{W},q}$ as the tree obtained from $\tau_{(S_1,S_2),q,\ell_1(q)}$, by appending, to every leaf $N' \in$ leaves$_{\tau_{(S_1,S_2),q,\ell_1(q)}}$ the tree $\tau_{(S_3,S_4),state(N'),e\text{-}level(N')}$.

LEMMA 4.12. *For every $\mathcal{U}$-witness $\mathcal{W}$ for (A, i, g) and every $q \in init(\mathcal{W})$, $\tau_{\mathcal{W},q}$ is an (i,g)-friendly (p,e)-tree rooted in q.*

4.4 From (p,e)-trees to witnesses

In this section, we first define a transformation from an (i,g)-friendly (p,e)-tree $\mathcal{T}$ rooted in q to a $\mathcal{U}$-witness $\mathcal{W}_{\mathcal{T}}$ for (A, i, g) with $q \in init(\mathcal{W}_{\mathcal{T}})$. Then, we show how to adapt such a transformation to suitably convert an i-invariant (p,e)-tree into a $\square$-witness.

At a very high level, we proceed as follows. First, we show how to obtain, from an (infinite) (i,g)-friendly (p,e)-tree, a finite A-tree which maintains enough significant information about the strategy represented by the original tree. From such a finite tree, we suitably choose nodes that are used as representatives for the vertices of the four partial witnesses that form the desired witness. Roughly speaking, each of these nodes define the behavior of a vertex in a pw by carrying information about its label (energy level) and outgoing transitions. A detailed outline of the transformation follows.

1. Build $\mathcal{T}'$. We obtain the finite A-tree $\mathcal{T}'$ from the infinite (p,e)-tree $\mathcal{T}$, by suitably cutting its branches. To this end, we identify the set of nodes of $\mathcal{T}$ whose descendant will be discharged to obtain $\mathcal{T}'$, or, in other words, the set of nodes that will be leaves in $\mathcal{T}'$, namely leaves$_{\mathcal{T}'}$. Since $\mathcal{T}$ is a (p,e)-tree, every branch $\mathcal{B}$ of $\mathcal{T}$ satisfies the parity condition, that is, there are along $\mathcal{B}$ infinitely many occurrences of a state, let us call it $q^{\mathcal{B}}$, such that $p(q^{\mathcal{B}})$ is even and $p(q^{\mathcal{B}}) \leq \min\{p(q'') \mid q''$ occurs infinitely often along $\mathcal{B}\}$; moreover, since $\mathcal{T}$ is (i,g)-friendly, $\mathcal{B}$ features at least one occurrence of a state from g. Thus, for every branch $\mathcal{B}$ of $\mathcal{T}$, there exists $N \in \mathcal{B}$ for which there are $N_1, N_2 \in \mathcal{B}$, with $N_2 \in ancestors(N)$ and $N_1 \in path\text{-}to(N_2)$, such that $state(N_1) \in g$, $state(N_2) = state(N) = q^{\mathcal{B}}$, $e\text{-}level(N_2) \leq e\text{-}level(N)$; let $N^{\mathcal{B}}$ be the earliest node in $\mathcal{B}$ meeting these conditions (i.e., no $N' \in ancestors(N^{\mathcal{B}})$ exists with the same properties). We define leaves$_{\mathcal{T}'} = \{N^{\mathcal{B}} \mid \mathcal{B}$ is a branch of $\mathcal{T}\}$ and $\mathcal{T}' = \mathcal{T} \setminus desc($leaves$_{\mathcal{T}'})$. By König's Lemma, $\mathcal{T}'$ is finite.

2. Build $\mathcal{T}''$. Analogously, we obtain the auxiliary finite tree $\mathcal{T}''$ from $\mathcal{T}'$ by suitably cutting its branches: we first define the set leaves$_{\mathcal{T}''}$ and then discharging descendants of nodes in such a set. Let leaves$_{\mathcal{T}''} = \{N \in \mathcal{T}' \mid state(N) \in g$ and $\forall N' \in ancestors(N). state(N') \notin g\}$; we define $\mathcal{T}'' = \mathcal{T}' \setminus desc($leaves$_{\mathcal{T}''})$.

Note that both $\mathcal{T}'$ and $\mathcal{T}''$ are A-trees. Moreover, we have that (i) $\mathcal{T}'' \subseteq \mathcal{T}' \subseteq \mathcal{T}$, (ii) $state(N) \in i$ for every $N \in \mathcal{T}'' \setminus$ leaves$_{\mathcal{T}''}$, and (iii) $state(N) \in g$ for every $N \in$ leaves$_{\mathcal{T}''}$.

3. Define a linear order $\prec$ over nodes in $\mathcal{T}' \setminus$ leaves$_{\mathcal{T}'}$. We fix a linear order $\prec$ over $\mathcal{T}' \setminus$ leaves$_{\mathcal{T}'}$ such that

$$\forall N, N' \in \mathcal{T}' \setminus \text{leaves}_{\mathcal{T}'} .\ \left(N' \in ancestors(N) \Rightarrow N \prec N'\right).$$

More precisely, $\prec$ can be thought of as any reverse topological ordering of nodes in $\mathcal{T}' \setminus$ leaves$_{\mathcal{T}'}$ (since $\mathcal{T}' \setminus$ leaves$_{\mathcal{T}'}$ is a DAG, a topological order over it exists).

4. Define two representative functions schemas. A *representative function* is a function $\mathrm{r} : V \to \mathcal{T}'$, where $V \subseteq Q$.

Let $\widehat{\mathcal{T}} \subseteq \mathcal{T}'$ be a set of nodes in $\mathcal{T}'$ and let $\prec$ be the linear order defined above. Moreover, let

$$V_{\widehat{\mathcal{T}}} = \{q \in Q \mid q = state(N) \text{ for some } N \in \widehat{\mathcal{T}}\}.$$

We define two representative functions schemas (parametric in $\widehat{\mathcal{T}}$).

- earliest$_{\widehat{\mathcal{T}}}$: $V_{\widehat{\mathcal{T}}} \to \mathcal{T}'$ s.t. for every $q \in V_{\widehat{\mathcal{T}}}$ earliest$_{\widehat{\mathcal{T}}}(q)$ is the earliest node $N \in \widehat{\mathcal{T}}$ (according to $\prec$) with state q; formally:
  (a) $state(N) = q$ and
  (b) $\forall N' \in \widehat{\mathcal{T}} .\ N' \prec N \Rightarrow state(N') \neq stateN$
- lowest-energy$_{\widehat{\mathcal{T}}}$: $V_{\widehat{\mathcal{T}}} \to \mathcal{T}'$ such that for every $q \in V_{\widehat{\mathcal{T}}}$ lowest-energy$_{\widehat{\mathcal{T}}}(q)$ is the earliest node $N \in \widehat{\mathcal{T}}$ (according to $\prec$) with state q and lowest energy level; formally:
  (a) $state(N) = q$,
  (b) $\forall N' \in \widehat{\mathcal{T}}$ with $state(N') = state(N)$
    – $N' \prec N \Rightarrow e\text{-}level(N') > e\text{-}level(N)$ and
    – $N \prec N' \Rightarrow e\text{-}level(N') \geq e\text{-}level(N)$.

5. Define function $\mathrm{pw}(\cdot,\cdot)$. We show how a set of vertices $V \subseteq Q$ and a representative function $\mathrm{r} : V \to \mathcal{T}'$ univocally identify a pw. Intuitively, the resulting pw has V as set of vertices; the behavior of each vertex $q \in V$, i.e., its label (function ℓ) and the way it evolves (relation T), is determined by $\mathrm{r}(q)$, its representative node in $\mathcal{T}'$.

Formally, let $V = V' \uplus V^{fin} \subseteq Q$ and $\mathrm{r} : V \to \mathcal{T}'$ be a representative function. The *pw for A induced by* $(V', V^{fin}, \mathrm{r})$ is (V, T, ℓ), where:

- T is the smallest set such that if $q \in V'$ and $N' \in children_{\mathcal{T}'}(\mathrm{r}(q))$, then $(q, in\text{-}action(N'), state(N')) \in T$, and
- $\ell(q) = e\text{-}level(\mathrm{r}(q))$ for every $q \in V$.

6. Build witness $\mathcal{W}_{\mathcal{T}} = (S_1, S_2, S_3, S_4)$. We show separately how to build (S_1, S_2) and (S_3, S_4).

6.1. Definition of S_1 and S_2. As a preliminary step, we build two auxiliary pw's for A:

- $S^{1,2}_{low} = (V^{1,2}_{low}, T^{1,2}_{low}, \ell^{1,2}_{low})$ is the pw induced by $(V_{\mathcal{T}''\setminus \text{leaves}_{\mathcal{T}''}}, V_{\text{leaves}_{\mathcal{T}''}}, \text{lowest-energy}_{\mathcal{T}''})$
- $S^{1,2}_{early} = (V^{1,2}_{early}, T^{1,2}_{early}, \ell^{1,2}_{early})$ is the pw induced by $(V_{\mathcal{T}''\setminus \text{leaves}_{\mathcal{T}''}}, V_{\text{leaves}_{\mathcal{T}''}}, \text{earliest}_{\mathcal{T}''})$.

Intuitively, vertices of $S^{1,2}_{low}$ are also vertices of S_1; in addition, vertices of $S^{1,2}_{low}$ occurring in some $S^{1,2}_{low}$-cycle also belong to S_2, along with their successors in $S^{1,2}_{low}$. Vertices of S_2 behave in S_2 as they do in $S^{1,2}_{low}$; vertices of S_1 not belonging to S_2 behave in S_1 as they do in $S^{1,2}_{low}$; vertices of S_1 also belonging to S_2 behave in S_1 as they do in $S^{1,2}_{early}$. Formally, components of $\mathcal{W}_{\mathcal{T}}$ are obtained as follows.

- $V'_1 = \{state(N) \mid N \in \mathcal{T}'' \setminus \text{leaves}_{\mathcal{T}''}\} \subseteq i$,
- $V^{fin}_1 = \{state(N) \mid N \in \text{leaves}_{\mathcal{T}''}\} \subseteq g$,
- $V_1 = V_{\mathcal{T}''} = V'_1 \cup V^{fin}$,
- *(vertices occurring in $S^{1,2}_{low}$-cycles and their successors in $S^{1,2}_{low}$ belong to S_2)* V_2 is the smallest set such that, for every $q \in V'_1$, if q occurs in an $S^{1,2}_{low}$-cycle then $\{q, q'\} \subseteq V_2$ for every $(q, \vec{\alpha}, q') \in T^{1,2}_{low}$,
- *(vertices of S_2 behave in S_2 as they do in $S^{1,2}_{low}$)*
  – $T_2 = \{(q, \vec{\alpha}, q') \in T^{1,2}_{low} \mid q \in V'_1$ and q occurs in an $S^{1,2}_{low}$-cycle$\}$,
  – $\ell_2(q) = \ell^{1,2}_{low}(q)$ for every $q \in V_2$,
- *(vertices of S_1 not belonging to S_2 behave in S_1 as they do in $S^{1,2}_{low}$)*
  – $T_1 \supseteq \{(q, \vec{\alpha}, q') \in T^{1,2}_{low} \mid q \in V'_1 \setminus V_2\}$,
  – $\ell_1(q) = \ell^{1,2}_{low}(q)$ for every $q \in V_1 \setminus V_2$,
- *(vertices of S_1 also belonging to S_2 behave in S_1 as they do in $S^{1,2}_{early}$)*
  – $T_1 \supseteq \{(q, \vec{\alpha}, q') \in T^{1,2}_{early} \mid q \in V'_1 \cap V_2\}$,

Alg. 2 In mixed instances, it checks for the existence of a (i, g, o)-friendly (p, e')-strategy for A from q, where $e' = \langle w, \mathcal{E}, [a, b]\rangle$.

procedure $\exists$-STRATEGY-MIXED$(\mathcal{G}, i, g, o, A, q, \mathcal{E})$
  if $o = \mathcal{U}$ **then** GUESS $\mathcal{W} = (S_1, S_2, S_3, S_4)$
  else GUESS $\mathcal{W} = (S_1, S_2)$
  if CHECK$(\mathcal{W}, o, A, i, g, q)$ **then return** TRUE ▷ checks if $\mathcal{W}$ is an o-witness ...
  return FALSE ▷ ...for (A, i, g) with $q \in init(\mathcal{W})$

- $\ell_1(q) = \ell^{1,2}_{early}(q)$ for every $q \in V_1 \cap V_2$,
- no other transition belongs to T_1.

6.2. Definition of S_3 and S_4. The definition of S_3 and S_4 is very similar to the one of S_1 and S_2, respectively, the only differences being (we let $\mathcal{X} = ((\mathcal{T}' \setminus \mathcal{T}'') \cup \text{leaves}_{\mathcal{T}''}) \setminus \text{leaves}_{\mathcal{T}'}$):

- functions lowest-energy$_{\mathcal{X}}$ and earliest$_{\mathcal{X}}$ replace lowest-energy$_{\mathcal{T}''}$ and earliest$_{\mathcal{T}''}$, respectively, as representative functions, and the two auxiliary pw's $S^{3,4}_{low}$ and $S^{3,4}_{early}$ for A are defined as:
  - $S^{3,4}_{low} = (V^{3,4}_{low}, T^{3,4}_{low}, \ell^{3,4}_{low})$ is the pw induced by $(V_{\mathcal{X}}, \emptyset, \text{lowest-energy}_{\mathcal{X}})$
  - $S^{3,4}_{early} = (V^{3,4}_{early}, T^{3,4}_{early}, \ell^{3,4}_{early})$ is the pw induced by $(V_{\mathcal{X}}, \emptyset, \text{earliest}_{\mathcal{X}})$.
- $V^{fin}_3 = \emptyset$ (whereas, possibly, $V^{fin}_1 \neq \emptyset$) and thus $V_3 = V'_3 = V_{\mathcal{X}}$,

It is possible to show the converse correspondence between witnesses and (p, e)-trees, as stated in the following lemma.

LEMMA 4.13. *If $\mathcal{T}$ is an (i, g)-friendly (p, e)-tree rooted in q, then $\mathcal{W}_{\mathcal{T}}$ is a $\mathcal{U}$-witness for (A, i, g) with $q \in init(\mathcal{W})$.*

The above procedure can be easily adapted as follows, to suitably convert i-invariant (p, e)-trees into $\Box$-witnesses for (A, i, g). By assuming $g = Q$, the procedure yields $\mathcal{W}_{\mathcal{T}} = (S_1, S_2, S_3, S_4)$, where A-accumulator (S_1, S_2) can be ignored (it is a trivial A-accumulator such that $S_1 = S_2$, $V_1 = \{state(\text{root}_{\mathcal{T}})\}$ and $T_1 = \emptyset$) as there is no need to search for goals; on the other hand, the A-accumulator (S_3, S_4) is the desired $\Box$-witness.

LEMMA 4.14. *If $\mathcal{T}$ is an i-invariant (p, e)-tree rooted in q, then $\mathcal{W}_{\mathcal{T}}$ is a $\Box$-witness for (A, i, g) with $q \in init(\mathcal{W})$.*

Finally, the next theorem immediately follows from Theorem 4.4, Corollary 4.11, Lemma 4.12, Lemma 4.13, and Lemma 4.14.

THEOREM 4.15. *An (i, g, o)-friendly (p,e)-strategy from q exists if and only if there is an o-witness $\mathcal{W}$ for (A, i, g) with $q \in init(\mathcal{W})$.*

Now, a simple algorithm (see Algorithm 2) to search for an $(i, g, \mathcal{U})$-friendly strategy non-deterministically guesses four pw's S_1, S_2, S_3, and S_4, and then checks that conditions *(a)-(c)* of Definition 4.7 are satisfied. Similarly, when looking for $(i, g, \Box)$-friendly strategies the algorithm non-deterministically guesses S_1 and S_2, and then checks that they form a $\Box$-*witness* (see Definition 4.7). Since such checks can be done in polynomial time, we have the following result.

THEOREM 4.16. *The procedure $\exists$-STRATEGY-MIXED runs in non-deterministic polynomial time in the size of the input.*

5 HARVEST AND CONCLUSIONS

We are finally ready to employ the results of the previous sections into a procedure for model checking pe-ATL, presented in Algorithm 3. The case of formulas of the kind $\langle\langle A\rangle\rangle \bigcirc \psi_1$ is worked out

Alg. 3 It solves the model checking problem for pe-ATL.

1: **procedure** PE-ATL-MC$(\mathcal{G}, q, \varphi)$
2: **if** $a \neq -\infty$ **and** $b \neq +\infty$ **then** x = bounded ▷ x is set to BOUNDED or MIXED ...
3: **if** $a \neq -\infty$ **and** $b = +\infty$ **then** x = mixed ▷ ...to suitably instantiate ...
4: **if** $[a, b] = [-\infty, +\infty]$ **then** ▷ ... $\exists$-STRATEGY-x as ...
5: $x \leftarrow$ bounded ▷ ... $\exists$-STRATEGY-BOUNDED or ...
6: $[a, b] \leftarrow [0, 0]$ ▷ ... $\exists$-STRATEGY-MIXED
7: $\mathcal{E}^{init} \leftarrow 0$ ▷ unbounded instances are ...
8: $w(q, \vec{\alpha}) \leftarrow 0$ for all $(q, \vec{\alpha})$ ▷ ...treated as special bounded ones
9: **for** $\psi \in Sub(\varphi)$ (increasingly ordered by size) **do**
10: $[\![\psi]\!] \leftarrow \emptyset$
11: **if** $\psi = \mathfrak{p}$ **then** $[\![\psi]\!] \leftarrow \pi(p)$ ▷ atomic propositon
12: **if** $\psi = \neg\psi_1$ **then** $[\![\psi]\!] \leftarrow Q \setminus [\![\psi_1]\!]$
13: **if** $\psi = \psi_1 \wedge \psi_2$ **then** $[\![\psi]\!] \leftarrow [\![\psi_1]\!] \cap [\![\psi_2]\!]$
14: **if** $\psi = \langle\langle A\rangle\rangle \bigcirc \psi_1$ **then**
15: **for** $q' \in Q$ **do**
16: **for** $\vec{\alpha}_A \in D_A(q)$ **do** ▷ cycle over proponent's strategies
17: good_strategy $\leftarrow$ TRUE
18: **for** $\vec{\alpha} \in D(q) \cap ext(\vec{\alpha}_A)$ **do** ▷ cycle over opponent's strategies
19: $q'' \leftarrow \delta(q', \vec{\alpha})$
20: $\mathcal{E} \leftarrow \mathcal{E}^{init} + w(q, \vec{\alpha})$
21: **if** $q'' \notin [\![\psi_1]\!]$ **or not** $\exists$-STRATEGY-$x(\mathcal{G}, Q, \emptyset, \Box, A, q'', \mathcal{E})$
22: **then** good_strategy $\leftarrow$ FALSE
23: **if** good_strategy **then** $[\![\psi]\!] \leftarrow [\![\psi]\!] \cup \{q\}$
24: **if** $\psi = \langle\langle A\rangle\rangle \psi_1 \mathcal{U} \psi_2$ **then**
25: **for** $q' \in Q$ **do**
26: **if** $\exists$-STRATEGY-$x(\mathcal{G}, [\![\varphi_1]\!], [\![\varphi_2]\!], \mathcal{U}, A, q', \mathcal{E}^{init})$ **then** $[\![\psi]\!] \leftarrow [\![\psi]\!] \cup \{q\}$
27: **if** $\psi = \langle\langle A\rangle\rangle \Box \psi_1$ **then**
28: **for** $q' \in Q$ **do**
29: **if** $\exists$-STRATEGY-$x(\mathcal{G}, [\![\varphi_1]\!], \emptyset, \Box, A, q', \mathcal{E}^{init})$ **then** $[\![\psi]\!] \leftarrow [\![\psi]\!] \cup \{q\}$
30: **return** $q \in [\![\varphi]\!]$

as one might expect; the only worthwhile remark is about line 22, where, towards the validation of a strategy, the algorithm checks that for all successors q'' of the state q' under investigation there is a (p,e)-strategy from q'' (besides checking that q''satisfy formula ψ_1): this is done by using the procedure $\exists$-STRATEGY-x (suitably instantiated depending on the type of instance in input) developed in the previous section to search for a $(Q, \emptyset, \Box)$-friendly (p,e)-strategy or, more intuitively, the algorithm checks that q'' satisfies formula $\langle\langle A\rangle\rangle\Box\top$. Formulas of kinds $\langle\langle A\rangle\rangle\psi_1\mathcal{U}\psi_2$ and $\langle\langle A\rangle\rangle\Box\psi_1$ are handled using the procedures described in the previous sections.

It is clear that complexity of the whole procedure is governed by the one of procedure $\exists$-STRATEGY-x ($x \in$ {BOUNDED, MIXED}). Thus, we have the following theorem, which comes from Theorem 3.4, Corollary 3.5, and Theorem 4.16.

THEOREM 5.1. *The model checking problem for pe-ATL is:*

- *in EXPTIME if $a, b \in \mathbb{Z}$ (bounded instances),*
- *in PTIME if $[a, b] = [-\infty, +\infty]$ (unbounded instances),*
- *in NP if $a \in \mathbb{Z}$ and $b = +\infty$ (mixed instances).*

The proposed setting follows a recent and promising trend devoted to the study of systems enabling qualitative and quantitative reasoning in MAS. Before the last decade, these two aspects have been mostly kept separate, despite their interplay in many natural application scenarios (e.g., allocation systems subject to energy constraints—see Introduction). Our proposal aims at developing a logical system able to deal with these two aspects jointly.

As future work, we aim at establishing thigh complexity bounds for the problems considered here, as well as considering different choices for modeling energy condition: at least another option is worth being considered, according to which energy level evolves while trying to satisfy the formula along the entire game (in our setting the energy level is reset whenever a new search for strategy by a possibly different team begins—see [4] for a comparison on the two approaches in the setting of ATL without parity condition).

REFERENCES

[1] N. Alechina, B. Logan, H.N. Nguyen, and F. Raimondi. 2017. Model-checking for Resource-Bounded ATL with production and consumption of resources. *J. Comput. Syst. Sci.* 88 (2017). https://doi.org/10.1016/j.jcss.2017.03.008

[2] R. Alur, T.A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *JACM* 49, 5 (2002).

[3] R. Bloem, K. Chatterjee, T.A. Henzinger, and B. Jobstmann. 2009. Better Quality in Synthesis through Quantitative Objectives. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings (Lecture Notes in Computer Science)*, Ahmed Bouajjani and Oded Maler (Eds.), Vol. 5643. 140–156. https://doi.org/10.1007/978-3-642-02658-4_14

[4] N. Bulling and B. Farwer. 2010. On the (Un-)Decidability of Model Checking Resource-Bounded Agents. In *Proc. of the 19th European Conference on Artificial Intelligence (ECAI) (Frontiers in Artificial Intelligence and Applications)*, Vol. 215. 567–572. https://doi.org/10.3233/978-1-60750-606-5-567

[5] N. Bulling and V. Goranko. 2013. How to Be Both Rich and Happy: Combining Quantitative and Qualitative Strategic Reasoning about Multi-Player Games (Extended Abstract). In *Proc. of the 1st International Workshop on Strategic Reasoning (SR) (EPTCS)*, Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi (Eds.), Vol. 112. 33–41. https://doi.org/10.4204/EPTCS.112.8

[6] C.F. Calvillo, A. Sánchez-Miralles, and J. Villar. 2016. Energy management and planning in smart cities. *Renewable and Sustainable Energy Reviews* 55 (2016).

[7] K. Chatterjee and L. Doyen. 2012. Energy parity games. *Theor. Comput. Sci.* 458 (2012). https://doi.org/10.1016/j.tcs.2012.07.038

[8] K. Chatterjee, T.A. Henzinger, and M. Jurdzinski. 2005. Mean-Payoff Parity Games. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*. 178–187. https://doi.org/10.1109/LICS.2005.26

[9] E.M. Clarke and E.A. Emerson. 1981. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *LP'81 (LNCS 131)*. 52–71.

[10] E.M. Clarke, O. Grumberg, and D.A. Peled. 2002. *Model Checking*. MIT Press.

[11] D. Della Monica, M. Napoli, and M. Parente. 2013. Model checking coalitional games in shortage resource scenarios. In *Proc. of the 4th International Symposium on Games, Automata, Logics and Formal Verification (GandALF)*. 240–255.

[12] N. Fijalkow and M. Zimmermann. 2014. Parity and Streett Games with Costs. *Logical Methods in Computer Science* 10, 2 (2014). https://doi.org/10.2168/LMCS-10(2:14)2014

[13] S. Ghosh and R. Ramanujam. 2011. Strategies in Games: A Logic-Automata Study. In *ESSLLI'11 (LNCS 7388)*. 110–159.

[14] J. Gutierrez, A. Murano, G. Perelli, S. Rubin, and M. Wooldridge. 2017. Nash Equilibria in Concurrent Games with Lexicographic Preferences. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). 1067–1073. https://doi.org/10.24963/ijcai.2017/148

[15] W. Jamroga and W. Penczek. 2011. Specification and Verification of Multi-Agent Systems. In *ESSLLI'11 (LNCS 7388)*. 210–263.

[16] W. Jamroga and W. van der Hoek. 2004. Agents that Know How to Play. *FI* 63, 2-3 (2004).

[17] O. Kupferman, M.Y. Vardi, and P. Wolper. 2001. Module Checking. *IC* 164, 2 (2001).

[18] F. Mogavero, A. Murano, and L. Sorrentino. 2015. On Promptness in Parity Games. *Fundam. Inform.* 139, 3 (2015). https://doi.org/10.3233/FI-2015-1235

[19] F. Mogavero, A. Murano, and M.Y. Vardi. 2010. Relentful Strategic Reasoning in Alternating-Time Temporal Logic. In *LPAR'10 (LNAI 6355)*. 371–387.

[20] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. 2015. Qualitative and quantitative monitoring of spatio-temporal properties. In *Runtime Verification*. Springer, 21–37.

[21] U. Zwick and M. Paterson. 1996. The Complexity of Mean Payoff Games on Graphs. *Theor. Comput. Sci.* 158, 1&2 (1996). https://doi.org/10.1016/0304-3975(95)00188-3