Relentful Strategic Reasoning in Alternating-Time Temporal Logic[1,2]

Fabio Mogavero, *Università degli Studi di Napoli Federico II, Napoli, Italy.*

Aniello Murano, *Università degli Studi di Napoli Federico II, Napoli, Italy.*

Moshe Y. Vardi, *Rice University, Houston, TX, U.S.A.*

Abstract

Temporal logics are a well investigated formalism for the specification, verification, and synthesis of reactive systems. Within this family, *Alternating-Time Temporal Logic* (ATL*, for short) has been introduced as a useful generalization of classical linear and branching-time temporal logics, by allowing temporal operators to be indexed by coalitions of agents. Classically, temporal logics are memoryless: once a path in the computation tree is quantified at a given node, the computation that has led to that node is forgotten. Recently, mCTL* has been defined as a memoryful variant of CTL*, where path quantification is memoryful. In the context of multi-agent planning, memoryful quantification enables agents to "relent" and change their goals and strategies depending on the histories of evolutions.

In this paper, we introduce *Relentful* ATL* (RATL*, for short), a kind of temporally-memoryful extension of ATL*, in which a formula is satisfied at a certain node of a play by taking into account both its future and past. We study the expressive power of RATL*, its succinctness, as well as related decision problems. We investigate the relationship between memoryful quantifications and past modalities and prove their equivalence. We also show that both the relentful and the past extensions come without any computational price; indeed, we prove that both the satisfiability and the model-checking problems are 2EXPTIME-COMPLETE, as for ATL*.

Keywords: Alternating-Time Temporal Logics, Backward Modalities, Strategic Reasoning, Game Logics

1 Introduction

Multi-agent concurrent systems recently emerged as a new paradigm for better understanding distributed systems [14, 50]. In such a formalization, different processes can interact in an adversarial or cooperative manner in order to achieve a number of possibly different goals. This setting can be seen as a multi-player game in the typical framework of game theory [39].

Classical branching-time temporal logics, such as CTL* [13], turn out to be of very limited power when applied to multi-agent systems. For example, consider the property $\mathtt{p}$: "processes 1 and 2 cooperate to ensure that a system (having more than two processes) never enters a failure state". It is well known that CTL* cannot express $\mathtt{p}$ [1]. Rather, CTL* can only say whether the set of all agents can or cannot prevent the system from failing.

In order to allow the temporal-logic framework to work within the setting of multi-agent systems, Alur, Henzinger, and Kupferman introduced *Alternating-Time Temporal Logic* (ATL*, for short) [1]. This is a generalization of CTL* obtained by replacing the path quantifiers, "E" (*there exists*) and "A" (*for all*), with "*cooperation modalities*" of the form $\langle\langle A \rangle\rangle$ and $[\![A]\!]$, where A is a set of *agents*. These modalities can be used to represent the power that a coalition

[1] Work supported in part by FP7 EU project 600958-SHERPA and POR Embedded System Cup B25B09090100007, by NSF grants CNS 1049862 and CCF-1139011, by NSF Expeditions in Computing project "ExCAPE: Expeditions in Computer Augmented Program Engineering", by BSF grant 9800096, and by gift from Intel.

[2] This work is partially based on the paper [35], which appeared in LPAR'10.

of agents has to achieve certain results. In particular, they can express selective quantifications over those paths that are obtained as outcomes of the infinite game between the coalition and its complement. ATL* formulas are interpreted over *concurrent game structures* (CGS, for short) [1], closely related to *systems* in [14], which model a set of interacting processes. Given a CGS $\mathcal{G}$ and a set A of agents, the ATL* formula $\langle\langle A\rangle\rangle\psi$ is satisfied at a state s of $\mathcal{G}$ iff there exists a *strategy* for the *team* A such that, no matter the strategy the *counterteam* $\mathrm{Ag} \setminus A$ executes, the resulting outcome of the interaction in $\mathcal{G}$ satisfies ψ at s. Coming back to the previous example, it is easy to see that the property p can be expressed by the ATL* formula $\langle\langle\{1,2\}\rangle\rangle\mathsf{G}\,\neg \mathit{fail}$, where G is the classic LTL temporal operator "*globally*".

Traditionally, temporal logics are *memoryless*: once a path in the underlying structure (usually a computation tree) is quantified at a given state, the computation that led to that state is forgotten [25]. In the case of ATL*, we have even more: the logic is also "relentless", in the sense that the agents are not able to formulate their strategies depending on the history of the computation; when $\langle\langle A\rangle\rangle\psi$ is asserted in a state s, its truth is independent of the path that led to s. Inspired by a work on *strong cyclic planning* [12], Pistore and Vardi proposed a logic that can express the spectrum between strong goal $\mathsf{A}\psi$ and the weak goal $\mathsf{E}\psi$ in planning [40]. A novel aspect of the Pistore-Vardi logic is that it is "*memoryful*", in the sense that the satisfiability of a formula at a state s depends on the future as well as on the past, *i.e.*, the trace starting from the initial state and leading to s. Nevertheless, this logic does not have a standard temporal logical syntax (for example, it is not closed under conjunction and disjunction). Also, it is less expressive than CTL*. This has lead Kupferman and Vardi [25] to introduce a memoryful variant of CTL* (mCTL*, for short), which unifies in a common framework both CTL* and the Pistore-Vardi logic. Syntactically, mCTL* is derived from CTL* by simply adding a special proposition $\mathsf{present}$, which is needed to emulate the ability of CTL* to talk about the "present" time. Semantically, mCTL* is obtained from CTL* by reinterpreting the path quantifiers of the logic to be memoryful.

Recently, ATL* has become a popular specification logic in the context of multi-agent system planning [19, 45, 18]. In such a framework, a temporally-memoryful enhancement of ATL*, in the spirit of the above discussion, would enable "*relentful*" planning. This means that, an agent can relent and change its goal, depending on its history[1]. In other words, when a specific goal at a certain state is checked, agents may learn from the past to change their goals. Note that this does not necessarily mean that agents change their strategy. As an example, consider the ATL* formula $[\![\mathrm{Ag}]\!]\,\mathsf{G}\,\langle\langle A\rangle\rangle\psi$. In the relentful framework, this formula would be satisfied by a CGS $\mathcal{G}$ (at a starting node s) iff agents in A can ensure that each possible trace (history) can be extended in an evolution of $\mathcal{G}$ satisfying ψ from s.

To give more insight on the relentful reasoning, consider a planning situation in which a team of agents wants to lay some objects on a table in a specific order (laying phase) and later to come back to the original situation (reverse phase). Assume that such a target can be represented by means of an LTL formula ψ. Also, suppose there is a configuration (*i.e.*, a specific distribution of the objects on the table) that is met during both the laying and the reverse phase. Clearly, by looking at this configuration without knowing the history, it is hard to understand if we are in the first or second phase. Consequently, it is not immediate to see how to formalize the strong cyclic planning specification related to the temporal goal ψ. In order to solve this ambiguity, one can adopt two different approaches. A syntactic solution would require injecting in the specification a considerable amount of information encoding the specific phase. A semantic one, instead, would allow to avoid the complication of the previous

[1] In Middle English to relent means to melt. In modern English it is used only in the combination of "relentless".

solution by exploiting a suitable relentful semantics of the strategic modalities.

In this paper, we introduce and study the logic RATL*, a relentful extension of ATL*. Thus, RATL* can be thought of as a fusion of mCTL* and ATL* in a common framework. Similarly to mCTL*, the syntax of RATL* is obtained from ATL* by simply adding a special proposition present. Semantically, RATL* is obtained from ATL* by reinterpreting the strategy quantifiers of the logic to be relentful, where present allows to identify the current moment. More specifically, for a CGS $\mathcal{G}$, the RATL* formula $\langle\langle A\rangle\rangle\psi$ holds at a state s of $\mathcal{G}$ if there is a strategy for agents in A such that, no matter which is the strategy of the agents not in A, the resulting outcome of the game, obtained by *extending* the execution trace of the system ending in s, satisfies ψ. Observe that, although the semantics of ATL* and RATL* are radically different, we can still express in the latter the classic ATL* specification $\langle\langle A\rangle\rangle\psi$, by exploiting the present proposition as follows: $\langle\langle A\rangle\rangle\mathsf{F}\,(\mathsf{present}\wedge\psi)$. Intuitively, we first reach the current moment in the time starting from the initial one and then verify ψ.

To show the usefulness of the relentful reasoning, consider as a synthetic example the situation in which the agents in a set A have the goal to eventually satisfy q and, if they see r, they can also change their goal to eventually satisfy v. It is easy to formalize this property in ATL* with the formula $\langle\langle A\rangle\rangle(\mathsf{F}\,q)\vee(\mathsf{F}\,r)\wedge(\mathsf{G}\,f)$, where f is $r\rightarrow\langle\langle A\rangle\rangle(\mathsf{F}\,v)$. Consider, instead, the situation in which the agents in A have the goal to satisfy p until q holds, unless they see r in which case they may change their goal to satisfy u until v holds. This is hard to be handled in ATL*, since the specification depends on the past due to the preconditions p and u. On the other hand, it can be easily specified in RATL* by means of the formula $\langle\langle A\rangle\rangle(p\,\mathsf{U}\,q)\vee(\mathsf{F}\,r)\wedge(\mathsf{G}\,f)$, where f is $r\rightarrow\langle\langle A\rangle\rangle(u\,\mathsf{U}\,v)$.

As a more concrete application of RATL*, consider the multi-agent planning scenario depicted in Figure 1, consisting of two robots, R_1 and R_2, that compete on placing some objects on their own pallets, P_1 and P_2, according to a specific goal. Apart from its own

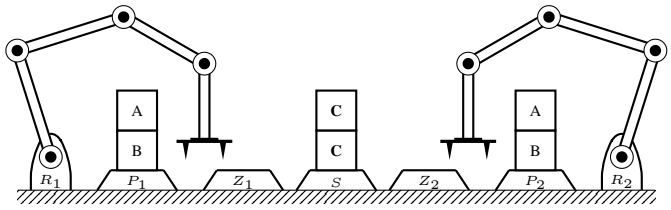

FIG. 1: A planning scenario in the block-world domain with two robots.

pallets, each robot R_i can access the auxiliary one Z_i. Moreover, both may access to a shared pallet S. The initial configuration comprises two C blocks piled on S and each P_i has a block A over a B one. The goal of each robot is to get as many C as possible; think of a situation in which the values of the blocks are ordered as follows: $A<B<C$. While the two robots are competing, it can happen that either *(i)* one of them gets both blocks C or *(ii)* only one of these blocks goes to each robot. In the first case, a backup goal for the unfortunate robot is to switch the order of its own blocks A and B; think of a penalty. In the second case, as it is considered a tie, the robots restore the original situation. As an additional argument, consider unpredictable events, as classically compelled in nondeterministic planning. In our specific example, they can refer to the fact that a robot fails to bring a block or that a block is contended between the two robots (see Figure 2). One can model this kind of unpredictability by means of a third agent N, which stand for *Nature*. The strong cyclic

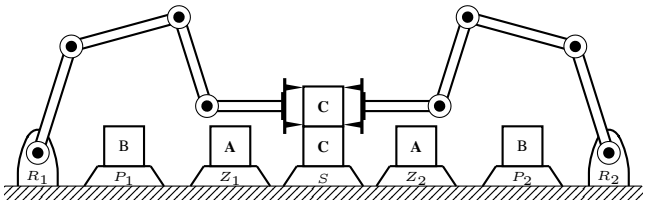

FIG. 2: Contention of a C block.

planning condition with respect to the above goals can be specified by means of the following RATL* formula $\varphi = [\![\{R_1, R_2, N\}]\!] \mathsf{G}\,(\bigwedge_{i=1}^{2} \bigvee_{j=1}^{3} \eta_i^j)$ with $\eta_i^1 = \langle\!\langle R_i, N\rangle\!\rangle \mathsf{F}\,\mathsf{G}\,\binom{C}{C}_i$, $\eta_i^2 = \langle\!\langle R_i, N\rangle\!\rangle \mathsf{F}\,\mathsf{G}\,\binom{B}{A}_i$, and $\eta_i^3 = \langle\!\langle R_i, N\rangle\!\rangle \mathsf{F}\,((\binom{A}{C}_i \vee \binom{B}{C}_i) \wedge \mathsf{F}\,\binom{A}{B}_i)$, where $\binom{X}{Y}_i$ is the atomic proposition representing the situation in which a block X is placed over a Y one on the pallet P_i. Specifically, φ ensures that, in each possible situation reached during a play, a robot R_i has the ability to enforce an evolution of the partial play up to that moment (the history) to achieve one of the three goals j previously described and formally represented by the subformula η_i^j, having the nature eventually behaving in a fair way. Observe that we cannot easily describe this property in ATL*, since the third goal η_i^3 is not prefix independent. Indeed, by looking only at the current state, we are not able to understand whether one of the two configurations $\binom{A}{C}_i$ and $\binom{B}{C}_i$ is already reached.

Along the paper, we also consider an extension of RATL* with *past operators* (RPATL*, for short). As for classical temporal and modal logics, past operators allow reasoning about the past in a computation [7, 6, 21, 22, 28, 29, 47]. In RPATL*, we can further require that coalitions of agents had a relentful goal in the past. In more details, we can write a formula whose satisfaction, at a state s, depends on the trace starting from the initial state and leading to a state s' occurring before s. Coming back to the previous synthetic example, by using P as the dual of F, we can change the alternative goal f of agents in A to be $r \rightarrow \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge \langle\!\langle A\rangle\!\rangle(u\,\mathsf{U}\,v))$, which requires that once r occurs at a given moment of a play, the subformula $\langle\!\langle A\rangle\!\rangle(u\,\mathsf{U}\,v)$ holds in the past, starting from the beginning of the computation which is reached by means of the apposite temporal operator $\tilde{\mathsf{Y}}\,\mathfrak{f}$.

As a direct consequence and important contribution of this work, we show for the first time a clear and complete picture of the relationships among ATL* and its various extensions with memoryful quantification and past modalities, which goes beyond the expressiveness results obtained in [25] for mCTL*. Since the relentful quantifiers refer to behaviors from the start of the computation, which occurred in the past, they are intimately connected to the past itself. Indeed, we prove this formally. We study the expressive power and the succinctness of RATL* with respect to ATL*, as well as the relentless fragment of RPATL* (*i.e.*, the extension of ATL* with past modalities), which we call PATL*. We show that the three logics have the same expressive power, but both RATL* and PATL* are at least exponentially more succinct than ATL*. As for R$^-$ATL* (where the minus stands for the variant of the logic without the present proposition, but the play interpretation is still relentful), we prove that it is strictly less expressive than ATL*. Conversely, we show that P$^-$ATL* is equivalent to PATL*, but exponentially less succinct.

From an algorithmic point of view, we examine, for RPATL*, the two classical decision problems: *model checking* and *satisfiability*. We show that model checking is not easier than satisfiability and in particular that are 2EXPTIME-COMPLETE, as for ATL*. We recall

that this is not the case for mCTL*, where the model checking is EXPSPACE-COMPLETE, while satisfiability is 2EXPTIME-COMPLETE. For the upper bounds, we follow an *automata-theoretic approach* [26]. In order to develop a decision procedure for a logic with the *tree-model property*, one first develops an appropriate notion of tree automata and studies their emptiness problem. Then, the decision problem for the logic can be reduced to the emptiness problem of such automata. To this aim, we introduce a new automaton model, namely the *symmetric alternating tree automata with satellites* (SATAS, for short), which extends both *automata over concurrent game structures* in [43] and *alternating automata with satellites* in [25], in a common setting. For technical convenience, the states of the whole automaton are partitioned into those regarding the satellite and the ones for the rest of the automaton, which we call the *main automaton*. The complexity results then come from the fact that RPATL* formulas can be translated into a SATAS with an exponential number of states for the main automaton and doubly exponential number of states for the satellite, and from the fact that the emptiness problem for this kind of automata is solvable in EXPTIME with respect to both the size of the main automaton and the logarithm of the size of the satellite.

Outline

In Section 2, we recall the basic notions regarding concurrent game structures and trees, tracks and paths, strategies, plays, and unwinding. Then, we have Section 3, in which we introduce RATL* by defining its syntax and semantics. In the following Section 4, we define the extension RPATL*, where the expressiveness and succinctness relationships of both the logics are studied. In Section 5, we introduce the SATAS automaton model. Finally, in Section 6, we describe how to solve the satisfiability and model-checking problems for both RATL* and RPATL*. Note that, in the accompanying Appendix A, we recall standard mathematical notation and some basic definitions that are used in the paper. Furthermore, in Appendix B, we introduce the full syntax and semantics of RPATL*.

2 Preliminaries

A *concurrent game structure* (CGS, for short) [1] is a tuple $\mathcal{G} \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, where AP and Ag are finite non-empty sets of *atomic propositions* and *agents*, Ac and St are enumerable non-empty sets of *actions* and *states*, $s_0 \in \mathrm{St}$ is a designated *initial state*, and $\lambda : \mathrm{St} \to 2^{\mathrm{AP}}$ is a *labeling function* that maps each state to the set of atomic propositions true in that state. Let $\mathrm{Dc} \triangleq \mathrm{Ac}^{\mathrm{Ag}}$ be the set of *decisions*, *i.e.*, functions from Ag to Ac representing the choices of an action for each agent. Then, $\tau : \mathrm{St} \times \mathrm{Dc} \to \mathrm{St}$ is a *transition function* mapping a pair of a state and a decision to a state. If the set of actions is finite, *i.e.*, $b = |\mathrm{Ac}| < \omega$, we say that $\mathcal{G}$ is b-bounded, or simply bounded. If both the sets of actions and states are finite, we say that $\mathcal{G}$ is finite.

Consider again the planning scenario of Figure 1. It can be simply formalized by means of a CGS $\mathcal{G}_P$ as follows. The atomic propositions in the set $\mathrm{AP} \triangleq \{\binom{X}{Y}_i : X, Y \in \mathrm{B} \wedge i \in \{1,2\} \wedge Y = \text{␣} \Rightarrow X = \text{␣}\}$ representing all situations in which a block $X \in \mathrm{B}$ is placed over a $Y \in \mathrm{B}$ one on the pallet P_i with $i \in \{1,2\}$, where $\mathrm{B} \triangleq \{A, B, C, \text{␣}\}$ is the set of possible blocks also containing the empty space ␣. Observe that the constraint $Y = \text{␣} \Rightarrow X = \text{␣}$ avoids to consider the propositions referring to the unfeasible situation of placing of a real block over ␣. The set of agents $\mathrm{Ag} \triangleq \{R_1, R_2, N\}$ just contains the two robots and the nature. At each moment of a play, R_1 and R_2 can either do nothing

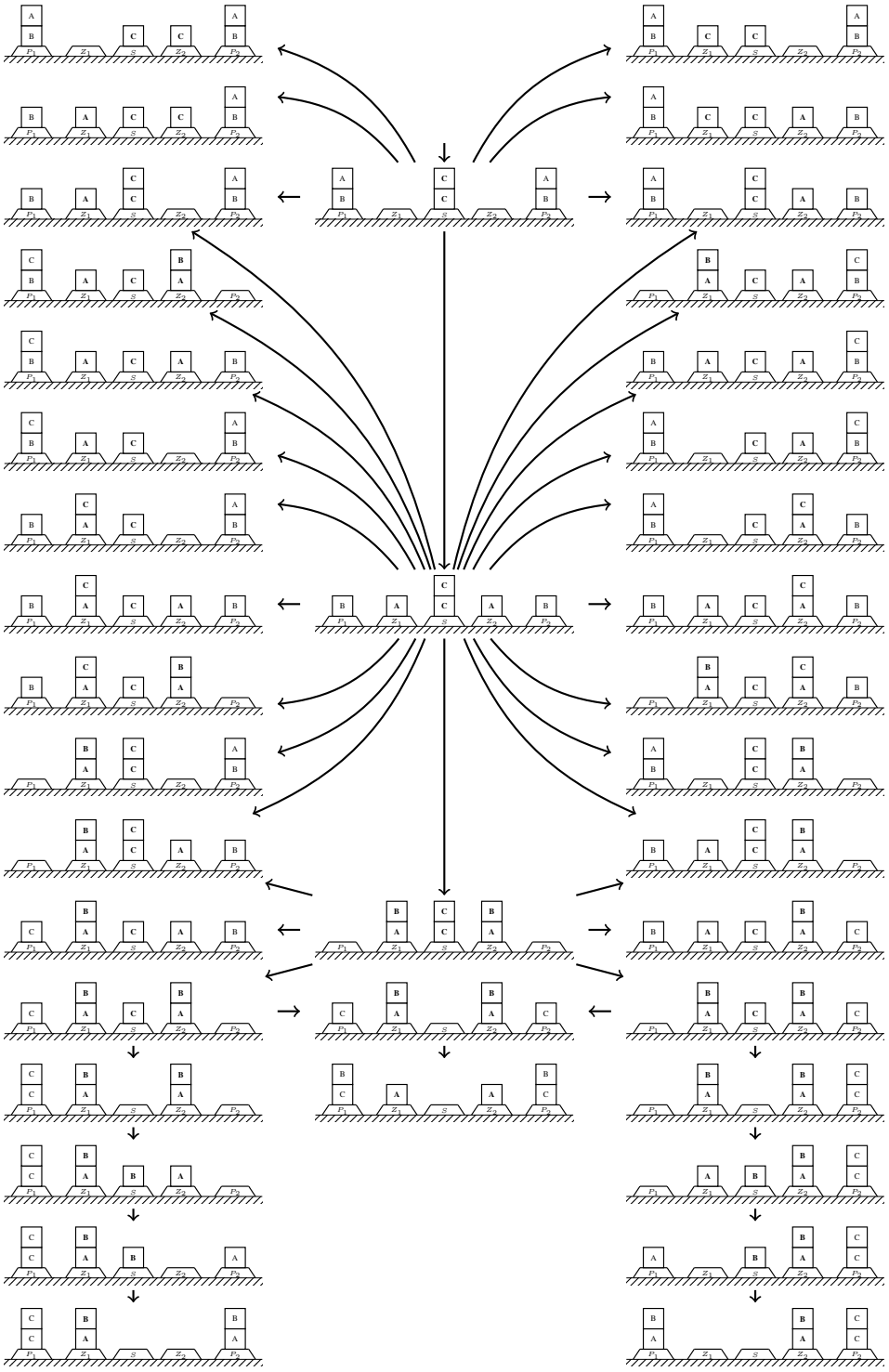

FIG. 3: Partial view on the state space in the planning scenario.

or transfer a block from a pallet to another one. Therefore, the related set of actions is $\mathrm{Ac}_R \triangleq \{I, P \mapsto Z, Z \mapsto P, P \mapsto S, S \mapsto P, Z \mapsto S, S \mapsto Z\}$, where I stands for the idle action, while $\alpha \mapsto \beta$ means that the block at the head of the stack on pallet α is moved to the stack on pallet β. The nature, instead, can either approve a move of a robot via the symbol $\checkmark$ or hinder its attempt via the symbol $\times$. Therefore, we set $\mathrm{Ac}_N \triangleq \{\checkmark, \times\} \times \{\checkmark, \times\}$, where the first component of the product is used to control the behavior of R_1, while the second one that of R_2. Summing up, we have $\mathrm{Ac} \triangleq \mathrm{Ac}_R \cup \mathrm{Ac}_N$. Note that, due to the uniformity of the CGS with respect to the usage of the actions by the agents, the robots can do a move also if it is not applicable to the current state. They can even use those of the nature, which can do the converse as well. As usual, these cumbersome situations can be suitable treated with opportune ending states. However, since our aim is to focus on the application of our logic, we avoid to deal with these non relevant facts. Every state $s \in \mathrm{St}$ of the game can be represented by means of a finite function $s : \mathrm{P} \to \mathrm{B} \times \mathrm{B}$ from the set of pallets $\mathrm{P} \triangleq \{P_1, P_2, Z_1, Z_2, S\}$ to pairs of blocks indicating the corresponding stack, with the proviso that there are exactly 2 occurrences of each real block. In particular, the first component of the pair is the head of the stack. Thus, the initial state is defined as follows: $s_0(P_i) \triangleq (A, B)$, $s_0(Z_i) \triangleq (\text{␣}, \text{␣})$, and $s_0(S) \triangleq (C, C)$, for $i \in \{1, 2\}$. Finally, we set $\lambda(s) \triangleq \{\binom{X}{Y}_i : i \in \{1, 2\} \wedge s(P_i) = (X, Y)\}$ as labeling function.

In Figure 3, we depict a small part of the underlying transition graph, where all self loops are omitted and the initial state s_0, placed in the top center position, is denoted by the entering edge without origin. To understand how to read this diagram, consider the situation in which, at s_0, both robots decide to take the action $P \mapsto Z$ and the nature is favorable to them by choosing the action $(\checkmark, \checkmark)$. Then, the system transits to the state s, with $s(P_i) = (\text{␣}, B)$, $s(Z_i) = (\text{␣}, A)$, and $s(S) = (C, C)$, at the center of the figure. If, instead, the nature wants to hinder the will of R_1 by choosing the action $(\times, \checkmark)$, the system transits to the state s, with $s(P_1) = (A, B)$, $s(P_2) = (\text{␣}, B)$, $s(Z_1) = (\text{␣}, \text{␣})$, $s(Z_2) = (\text{␣}, A)$, and $s(S) = (C, C)$, just on the right of the initial state. Suppose now that, at the central state, the two robots simultaneously decide to take a C block from the shared pallet by taking the action $S \mapsto Z$. In this case, if the nature is either favorable or wants to hinder both of them, the state does not change. Otherwise, in dependence of the action $(\checkmark, \times)$ or $(\times, \checkmark)$, the system transits either to the state s on the left, with $s(P_i) = (\text{␣}, B)$, $s(Z_1) = (C, A)$, $s(Z_2) = (\text{␣}, A)$, and $s(S) = (\text{␣}, C)$, or to the one on the right, with $s(P_i) = (\text{␣}, B)$, $s(Z_1) = (\text{␣}, A)$, $s(Z_2) = (C, A)$, and $s(S) = (\text{␣}, C)$.

Given a set $A \subseteq \mathrm{Ag}$ of agents, a *decision* and a *counterdecision* for A are, respectively, two functions $\mathsf{d}_A \in \mathrm{Ac}^A$ and $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag}\setminus A}$. By $\mathsf{d} \triangleq (\mathsf{d}_A, \mathsf{d}^c_A) \in \mathrm{Dc}$ we denote the *composition* of d_A and d^c_A, *i.e.*, the total decision such that $\mathsf{d}_{\restriction A} = \mathsf{d}_A$ and $\mathsf{d}_{\restriction (\mathrm{Ag}\setminus A)} = \mathsf{d}^c_A$.

A *track* (resp., *path*) in a CGS $\mathcal{G}$ is a finite (resp., an infinite) sequence of states $\rho \in \mathrm{St}^*$ (resp., $\pi \in \mathrm{St}^\omega$) such that, for all $i \in [0, |\rho| - 1[$ (resp., $i \in \mathbb{N}$), there exists a decision $\mathsf{d} \in \mathrm{Dc}$ such that $(\rho)_{i+1} = \tau((\rho)_i, \mathsf{d})$ (resp., $(\pi)_{i+1} = \tau((\pi)_i, \mathsf{d})$). A track ρ is *non-trivial* if $|\rho| > 0$, *i.e.*, $\rho \neq \varepsilon$. $\mathrm{Trk} \subseteq \mathrm{St}^+$ (resp., $\mathrm{Pth} \subseteq \mathrm{St}^\omega$) denotes the set of all non-trivial tracks (resp., paths). Moreover, $\mathrm{Trk}(s) \triangleq \{\rho \in \mathrm{Trk} : \mathsf{fst}(\rho) = s\}$ (resp., $\mathrm{Pth}(s) \triangleq \{\pi \in \mathrm{Pth} : \mathsf{fst}(\pi) = s\}$) indicates the subsets of tracks (resp., paths) starting at a state $s \in \mathrm{St}$.

A *strategy* for $\mathcal{G}$ with respect to a set of agents $A \subseteq \mathrm{Ag}$ is a partial function $\mathsf{f}_A : \mathrm{Trk} \rightharpoonup \mathrm{Ac}^A$ that maps a non-empty trace ρ in its domain to a decision $\mathsf{f}_A(\rho)$ of agents in A. Intuitively, a strategy for agents in A is a *combined plan* that contains all choices of moves as a function of the history of the current outcome. For a state s, we say that f_A is *s-total* iff it is defined on all non-trivial tracks starting in s that are reachable through f_A itself, *i.e.*, $\rho \cdot s' \in \mathsf{dom}(\mathsf{f}_A)$, with $\rho \in \mathsf{dom}(\mathsf{f}_A)$, iff $\mathsf{fst}(\rho) = s$ and there is a counterdecision $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag}\setminus A}$ for A such that

$\tau(\mathsf{lst}(\rho), (\mathsf{f}_A(\rho), \mathsf{d}^c_A)) = s'$. We use $\mathrm{Str}(A)$ (resp., $\mathrm{Str}(A, s)$ with $s \in \mathrm{St}$) to indicate the set of all strategies (resp., s-total strategies) of agents in A.

A path π in $\mathcal{G}$ starting at a state s is a *play* with respect to an s-total strategy f_A (f_A-*play*, for short) iff, for all $i \in \mathbb{N}$, there is a counterdecision $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag}\setminus A}$ such that $\pi_{i+1} = \tau(\pi_i, \mathsf{d})$, where $\mathsf{d} = (\mathsf{f}_A(\pi_{\leq i}), \mathsf{d}^c_A)$. Observe that π is an f_A-play iff $\pi_{\leq i} \in \mathsf{dom}(\mathsf{f}_A)$, for all $i \in \mathbb{N}$. Intuitively, a play is the outcome of the game determined by all the agents participating to it. By $\mathrm{Play}(\mathsf{f}_A)$ we denote the set of all f_A-plays.

A *concurrent game tree* (CGT, for short) is a CGS $\mathcal{T} \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, \varepsilon \rangle$, where *(i)* $\mathrm{St} \subseteq \Delta^*$ is a Δ-tree for a given set Δ of directions and *(ii)* if $t \cdot e \in \mathrm{St}$ then there is a decision $\mathsf{d} \in \mathrm{Dc}$ such that $\tau(t, \mathsf{d}) = t \cdot e$, for all $t \in \mathrm{St}$ and $e \in \Delta$. Furthermore, $\mathcal{T}$ is a *decision tree* (DT, for short) if *(i)* $\mathrm{St} = \mathrm{Dc}^*$ and *(ii)* if $t \cdot \mathsf{d} \in \mathrm{St}$ then $\tau(t, \mathsf{d}) = t \cdot \mathsf{d}$, for all $t \in \mathrm{St}$ and $\mathsf{d} \in \mathrm{Dc}$.

Given a CGS $\mathcal{G}$, its *unwinding* is the DT $\mathcal{G}_U \triangleq \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{Dc}^*, \lambda', \tau', \varepsilon \rangle$ for which there is a surjective function $\mathsf{unw} : \mathrm{Dc}^* \to \mathrm{St}$ such that *(i)* $\mathsf{unw}(\varepsilon) = s_0$, *(ii)* $\mathsf{unw}(\tau'(t, \mathsf{d})) = \tau(\mathsf{unw}(t), \mathsf{d})$, and *(iii)* $\lambda'(t) = \lambda(\mathsf{unw}(t))$, for all $t \in \mathrm{Dc}^*$ and $\mathsf{d} \in \mathrm{Dc}$.

From now on, we use the name of a CGS as a subscript to extract the components from its tuple-structure. Accordingly, if $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, we have $\mathrm{Ac}_\mathcal{G} = \mathrm{Ac}$, $\lambda_\mathcal{G} = \lambda$, $s_{0\mathcal{G}} = \mathsf{s}_0$, and so on. Also, we use the same notational concept to make explicit to which CGS the sets Dc, Trk, Pth, etc. are related to. Note that, we omit the subscripts if the structure can be unambiguously individuated from the context.

3 Relentful Alternating-Time Temporal Logic

In this section, we introduce an extension of classic alternating-time temporal logic ATL* [1], obtained by allowing the use of relentful quantification over plays, in a similar way it has been done for the memoryful branching-time temporal logic mCTL* over paths [25].

3.1 Syntax

The *relentful alternating-time temporal logic* (RATL*, for short) inherits from ATL* the existential $\langle\!\langle A \rangle\!\rangle$ and the universal $[\![A]\!]$ *strategy quantifiers*, where A denotes a set of agents. We recall that these two quantifiers can be read as *"there exists a collective strategy for agents in A"* and *"for all collective strategies of agents in A"*, respectively. The syntax of RATL* is similar to that for ATL*: there are two types of formulas, *state* and *path formulas*. Strategy quantifiers can prefix an assertion composed of an arbitrary Boolean combination and nesting of the linear-time operators X *"next"*, U *"until"*, and R *"release"*. The only syntactical difference between the two logics is that RATL* formulas can refer to a special proposition present, which enables us to refer to the present moment in the time. Readers familiar with mCTL* can see RATL* as mCTL* where strategy quantifiers substitute path quantifiers. The formal syntax of RATL* follows.

DEFINITION 3.1 (RATL* Syntax)
RATL* *state* (φ) and *path* (ψ) *formulas* are built inductively from the sets of atomic propositions AP and agents Ag in the following way, where $p \in \mathrm{AP}$ and $A \subseteq \mathrm{Ag}$:

1. $\varphi ::= \mathsf{present} \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\!\langle A \rangle\!\rangle \psi \mid [\![A]\!] \psi$;
2. $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi \mid \psi\,\mathsf{R}\,\psi$.

RATL* is the set of all state formulas generated by the above grammar, in which the occurrences of the special proposition present is in the scope of a strategy quantifier.

We now introduce some auxiliary syntactical notation.

For a formula φ, we define the *length* $\mathsf{lng}(\varphi)$ of φ as for ATL* [1]. Formally, *(i)* $\mathsf{lng}(p) \triangleq 1$, for $p \in \mathrm{AP} \cup \{\mathsf{present}\}$, *(ii)* $\mathsf{lng}(\mathsf{Op}\,\psi) \triangleq 1 + \mathsf{lng}(\psi)$, for all $\mathsf{Op} \in \{\neg, \mathsf{X}\}$, *(iii)* $\mathsf{lng}(\psi_1 \mathsf{Op}\, \psi_2) \triangleq 1 + \mathsf{lng}(\psi_1) + \mathsf{lng}(\psi_2)$, for all $\mathsf{Op} \in \{\wedge, \vee, \mathsf{U}, \mathsf{R}\}$, and *(iv)* $\mathsf{lng}(\mathsf{Qn}\,\psi) \triangleq 1 + \mathsf{lng}(\psi)$, for all $\mathsf{Qn} \in \{\langle\langle A\rangle\rangle, [\![A]\!]\}$.

We also use $\mathsf{cl}(\psi)$ to denote a variation of the classical Fischer-Ladner *closure* [15] of ψ defined recursively as for ATL* in the following way: *(i)* $\mathsf{cl}(p) \triangleq \emptyset$, for $p \in \mathrm{AP} \cup \{\mathsf{present}\}$, *(ii)* $\mathsf{cl}(\mathsf{Op}\,\psi) \triangleq \mathsf{cl}(\psi)$, for all $\mathsf{Op} \in \{\neg, \mathsf{X}\}$, *(iii)* $\mathsf{cl}(\psi_1 \mathsf{Op}\, \psi_2) \triangleq \mathsf{cl}(\psi_1) \cup \mathsf{cl}(\psi_2)$, for all $\mathsf{Op} \in \{\wedge, \vee, \mathsf{U}, \mathsf{R}\}$, and *(iv)* $\mathsf{cl}(\mathsf{Qn}\,\psi) \triangleq \{\mathsf{Qn}\,\psi\} \cup \mathsf{cl}(\psi)$, for all $\mathsf{Qn} \in \{\langle\langle A\rangle\rangle, [\![A]\!]\}$. Intuitively, $\mathsf{cl}(\varphi)$ is the set of all *basic formulas* that are subformulas of φ. Finally, by $\mathsf{rcl}(\psi)$ we denote the *reduced closure* of ψ, *i.e.*, the set of maximal basic formulas contained in ψ. Formally, (i) $\mathsf{rcl}(\varphi) \triangleq \{\varphi\}$, for all basic formulas $\varphi = \mathsf{Qn}\,\psi$, with $\mathsf{Qn} \in \{\langle\langle A\rangle\rangle, [\![A]\!]\}$, *(ii)* $\mathsf{rcl}(\mathsf{Op}\,\psi) \triangleq \mathsf{rcl}(\psi)$ when $\mathsf{Op}\,\psi$ is a path formula, for all $\mathsf{Op} \in \{\neg, \mathsf{X}\}$, and *(iii)* $\mathsf{rcl}(\psi_1 \mathsf{Op}\, \psi_2) \triangleq \mathsf{rcl}(\psi_1) \cup \mathsf{rcl}(\psi_2)$ when $\psi_1 \mathsf{Op}\, \psi_2$ is a path formula, for all $\mathsf{Op} \in \{\wedge, \vee, \mathsf{U}, \mathsf{R}\}$. It is immediate to see that $\mathsf{rcl}(\psi) \subseteq \mathsf{cl}(\psi)$ and $|\mathsf{cl}(\psi)| = \mathtt{O}(\mathsf{lng}(\psi))$.

3.2 Semantics

As for ATL*, the semantics of RATL* is defined with respect to concurrent game structures. However, the two logics differ on interpreting state formulas. First, in RATL* the satisfaction of a state formula is related to a specific track, while in ATL* it is related only to a state. Moreover, a path quantification in RATL* ranges over paths that start at the initial state and contain as prefix the track that lead to the present state. We refer to this track as the *present track*. The whole concept is what we name *relentful quantification*. On the contrary, in ATL*, path quantifications range over paths that start at the present state. For example, consider the formula $\varphi = [\![A]\!]\mathsf{G}\,\langle\langle B\rangle\rangle\psi$. Considered as an ATL* formula, φ holds in the initial state of a structure if the agents in B can force a path satisfying ψ from every state that can be reached by a strategy of the agents in A. In contrast, considered as an RATL* formula, φ holds in the initial state of the structure if the agents in B can extend to a path satisfying ψ every track generated by a strategy of the agents in A. Thus, when evaluating path formulas in RATL* one cannot ignore the past, and satisfaction may depend on the events that preceded the point of quantification. In ATL*, state and path formulas are evaluated with respect to states and paths in the structure, respectively. In RATL*, instead, we add an additional parameter, the *present track*, which is the track that lead from the initial state to the point of quantification. Path formulas are again evaluated with respect to paths, but state formulas are now evaluated with respect to tracks, which are viewed as partial executions.

We now formally define RATL* semantics with respect to a CGS $\mathcal{G}$. For two non-empty initial tracks $\rho, \rho_p \in \mathrm{Trk}(s_0)$, where ρ_p is the present track, we write $\mathcal{G}, \rho, \rho_p \models \varphi$ to indicate that the state formula φ holds at ρ, with ρ_p being the present. Similarly, for a path $\pi \in \mathrm{Pth}(s_0)$, a non-empty present track $\rho_p \in \mathrm{Trk}(s_0)$ and a natural number k, we write $\mathcal{G}, \pi, k, \rho_p \models \psi$ to indicate that the path formula ψ holds at the position k of π, with ρ_p denoting the present moment. The semantics of RATL* state formulas involving $\neg$, $\wedge$, and $\vee$, as well as that for RATL* path formulas, except for the state formula case, is defined as usual in ATL*. The semantics of the remaining part, which involves the relentful feature, follows.

DEFINITION 3.2 (RATL* Semantics)
Given a CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, two initial traces $\rho, \rho_p \in \mathrm{Trk}(s_0)$, a path $\pi \in \mathrm{Pth}(s_0)$, and a number $k \in \mathbb{N}$, it holds that:

1. $\mathcal{G}, \rho, \rho_p \models \mathsf{present}$ if $\rho = \rho_p$;
2. $\mathcal{G}, \rho, \rho_p \models p$ if $p \in \lambda(\mathsf{lst}(\rho))$, with $p \in \mathrm{AP}$;
3. $\mathcal{G}, \rho, \rho_p \models \langle\!\langle A \rangle\!\rangle \psi$ if there exists a $\mathsf{lst}(\rho)$-total strategy $\mathsf{f}_A \in \mathrm{Str}(A, \mathsf{lst}(\rho))$ such that, for all plays $\pi \in \mathrm{Play}(\mathsf{f}_A)$, it holds that $\mathcal{G}, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$;
4. $\mathcal{G}, \rho, \rho_p \models [\![A]\!] \psi$ if, for all $\mathsf{lst}(\rho)$-total strategies $\mathsf{f}_A \in \mathrm{Str}(A, \mathsf{lst}(\rho))$, there exists a play $\pi \in \mathrm{Play}(\mathsf{f}_A)$ such that $\mathcal{G}, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$;
5. $\mathcal{G}, \pi, k, \rho_p \models \varphi$ if $\mathcal{G}, \pi_{\leq k}, \rho_p \models \varphi$.

Observe that the present track ρ_p is used in the above definition only at Item 1 and that formulas of the form $\langle\!\langle A \rangle\!\rangle \psi$ and $[\![A]\!] \psi$ "set the new present", *i.e.*, their satisfaction with respect to ρ and ρ_p is independent of ρ_p, and the present trace, for the path formula ψ, is set to ρ.

Let $\mathcal{G}$ be a CGS and φ be an RATL* formula. Then, $\mathcal{G}$ is a *model* for φ, in symbols $\mathcal{G} \models \varphi$, iff $\mathcal{G}, s_0, s_0 \models \varphi$, where we recall that s_0 is the initial state of $\mathcal{G}$. In this case, we also say that $\mathcal{G}$ is a model for φ on s_0. A formula φ is said *satisfiable* iff there exists a model for it. Moreover, it is an *invariant* for the two CGSs $\mathcal{G}_1$ and $\mathcal{G}_2$ iff either $\mathcal{G}_1 \models \varphi$ and $\mathcal{G}_2 \models \varphi$ or $\mathcal{G}_1 \not\models \varphi$ and $\mathcal{G}_2 \not\models \varphi$.

For all state formulas φ_1 and φ_2, we say that φ_1 *implies* φ_2, in symbols $\varphi_1 \Rightarrow \varphi_2$, iff, for all CGS $\mathcal{G}$ and non-empty traces $\rho, \rho_p \in \mathrm{Trk}(\mathcal{G}, s_0)$, it holds that if $\mathcal{G}, \rho, \rho_p \models \varphi_1$ then $\mathcal{G}, \rho, \rho_p \models \varphi_2$. Consequently, we say that φ_1 is *equivalent* to φ_2, in symbols $\varphi_1 \equiv \varphi_2$, iff both $\varphi_1 \Rightarrow \varphi_2$ and $\varphi_2 \Rightarrow \varphi_1$ hold.

W.l.o.g., in the rest of the paper, we mainly consider formulas in *existential normal form* (*enf*, for short), *i.e.*, only existential strategy quantifiers occur. Indeed, all formulas can be linearly translated in *enf* by using De Morgan's laws together with the following equivalences, which directly follow from the semantics of the logic: $\neg \mathsf{X}\, \varphi \equiv \mathsf{X}\, \neg\varphi$, $\neg(\varphi_1 \mathsf{U}\, \varphi_2) \equiv (\neg\varphi_1) \mathsf{R}\, (\neg\varphi_2)$, and $\neg\langle\!\langle x \rangle\!\rangle \varphi \equiv [\![x]\!] \neg\varphi$.

As an example of the usefulness of RATL*, consider now the CGS $\mathcal{G}_P$ of Figure 3 for the planning situation of Figure 1, together with the previously described RATL* specification $\varphi = [\![\{R_1, R_2, N\}]\!] \mathsf{G}\, (\bigwedge_{i=1}^{2} \bigvee_{j=1}^{3} \eta_i^j)$ with $\eta_i^1 = \langle\!\langle R_i, N \rangle\!\rangle \mathsf{F}\, \mathsf{G}\, \binom{C}{C}_i$, $\eta_i^2 = \langle\!\langle R_i, N \rangle\!\rangle \mathsf{F}\, \mathsf{G}\, \binom{B}{A}_i$, and $\eta_i^3 = \langle\!\langle R_i, N \rangle\!\rangle \mathsf{F}\, ((\binom{A}{C}_i \vee \binom{B}{C}_i) \wedge \mathsf{F}\, \binom{A}{B}_i)$. We recall that the aim of the game for each robot is to collect as many C blocks as possible. In case both robots just get one C, it is a tie, so they have to come back to the original configuration on their own pallets P. Conversely, if one robot does not obtain a C, it loses the game, so it has to pay a penalty by switching its own blocks A and B. Now, it is not hard to see that $\mathcal{G}_P \models \varphi$. Indeed, at each moment of a given play, the two robots have the possibility to continue it, in a way to verify one of the three requirements, once the nature does not persist in hindering their attempts. For example, consider the situation in which an history of a play, *i.e.*, a track, has reached the state s, with $s(P_i) = (B, C)$, $s(Z_i) = (\text{␣}, A)$, and $s(S) = (\text{␣}, \text{␣})$, at the lower middle place of Figure 3. It is evident that the play is a tie, so, they have to restore the initial positions of the blocks A and B, by simply undo their moves. Note that, to express the same property in ATL*, one has to heavily modify the formula φ in order to insert all the information needed to decide whether a given state, such as the one just above the former, is reached either before or after the verification of a tie situation. At the end of the next section, we show why this is actually the case, by describing the translation of a simple RATL* formula similar to φ.

By induction on the syntactical structure of the sentences, it is easy to prove the following two classical results. Note that these are the basic steps towards the automata-theoretic approach we use to solve the model-checking and the satisfiability problems for RATL*.

THEOREM 3.3 (RATL* Unwinding Invariance)
RATL* is invariant under unwinding, *i.e.*, for each CGS $\mathcal{G}$ and formula φ, it holds that φ is an invariant for $\mathcal{G}$ and $\mathcal{G}_U$.

PROOF. Preliminary, let $\mathsf{unw}_{trk} : \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon) \to \mathrm{Trk}_{\mathcal{G}}(s_{0\mathcal{G}})$ and $\mathsf{unw}_{pth} : \mathrm{Pth}_{\mathcal{G}_U}(\varepsilon) \to \mathrm{Pth}_{\mathcal{G}}(s_{0\mathcal{G}})$ be the two functions mapping tracks and paths of the unwinding $\mathcal{G}_U$ into the corresponding ones of the original model $\mathcal{G}$, which satisfy the following properties: *(i)* $\mathsf{unw}_{trk}(\varepsilon) = s_{0\mathcal{G}}$, *(ii)* $\mathsf{unw}_{trk}(\rho \cdot t) = \mathsf{unw}_{trk}(\rho) \cdot \mathsf{unw}(t)$, for all $\rho \cdot t \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$ with $t \in \mathrm{St}_{\mathcal{G}_U}$, and *(iii)* $(\mathsf{unw}_{pth}(\pi))_{\leq i} = \mathsf{unw}_{trk}((\pi)_{\leq i})$, for all $\pi \in \mathrm{Pth}_{\mathcal{G}_U}(\varepsilon)$ and $i \in \mathbb{N}$. Note that $\varepsilon \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$ is not the empty track, but the track of length 1 made by the root of the tree only. Moreover, consider the following orderings between tracks and paths of $\mathcal{G}_U$: *(i)* $\rho < \rho'$ iff there exists a track $\rho'' \in \mathrm{Trk}_{\mathcal{G}_U}$ such that $\rho' = \rho \cdot \rho''$, for all $\rho, \rho' \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$; *(ii)* $\rho < \pi$ iff there exists a path $\pi' \in \mathrm{Pth}_{\mathcal{G}_U}$ such that $\pi = \rho \cdot \pi'$, for all $\rho \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$ and $\pi \in \mathrm{Pth}_{\mathcal{G}_U}(\varepsilon)$. Observe that $<$ forms a strict partial order on tracks.

To prove the statement, we show that, for all state formulas φ and path formulas ψ, it holds that *(i)* $\mathcal{G}_U, \rho, \rho_p \models \varphi$ iff $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models \varphi$, for all $\rho, \rho_p \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$, such that either $\rho < \rho_p$ or $\rho = \rho_p$ or $\rho_p < \rho$, and *(ii)* $\mathcal{G}_U, \pi, k, \rho_p \models \psi$ iff $\mathcal{G}, \mathsf{unw}_{pth}(\pi), k, \mathsf{unw}_{trk}(\rho_p) \models \psi$, for all $\pi \in \mathrm{Pth}_{\mathcal{G}_U}(\varepsilon)$, $k \in SetN$, and $\rho_p \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$, such that $\rho_p < \pi$.

By induction on the structure of formulas, we show the three cases of special proposition $\mathsf{present}$, atomic proposition p, and existential quantifier $\langle\langle \mathrm{A} \rangle\rangle \psi$. The remaining cases are immediate or easily derivable by the former ones.

- $(\varphi = \mathsf{present})$
  By the semantics definition, we have that $\mathcal{G}_U, \rho, \rho_p \models \mathsf{present}$ iff $\rho = \rho_p$ and $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models \mathsf{present}$ iff $\mathsf{unw}_{trk}(\rho) = \mathsf{unw}_{trk}(\rho_p)$. Now, by the hypothesis $\rho < \rho_p$ or $\rho = \rho_p$ or $\rho_p < \rho$ on the tracks ρ and ρ_p, we have that $\rho = \rho_p$ iff $\mathsf{unw}_{trk}(\rho) = \mathsf{unw}_{trk}(\rho_p)$. Therefore, $\mathcal{G}_U, \rho, \rho_p \models \mathsf{present}$ iff $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models \mathsf{present}$.
- $(\varphi = p)$
  By the definition of unw_{trk}, we have that $\mathsf{lst}(\mathsf{unw}_{trk}(\rho)) = \mathsf{unw}(\mathsf{lst}(\rho))$. Consequently, by the definition of unwinding function unw, it holds that $\lambda_{\mathcal{G}}(\mathsf{lst}(\mathsf{unw}_{trk}(\rho))) = \lambda_{\mathcal{G}_U}(\mathsf{lst}(\rho))$. Thus, we derive that $\mathcal{G}_U, \rho, \rho_p \models p$ iff $p \in \lambda_{\mathcal{G}_U}(\mathsf{lst}(\rho))$ iff $p \in \lambda_{\mathcal{G}}(\mathsf{lst}(\mathsf{unw}_{trk}(\rho)))$ iff $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models p$. Hence, $\mathcal{G}_U, \rho, \rho_p \models p$ iff $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models p$.
- $(\varphi = \langle\langle \mathrm{A} \rangle\rangle \psi, \Rightarrow)$
  Suppose that $\mathcal{G}_U, \rho, \rho_p \models \langle\langle \mathrm{A} \rangle\rangle \psi$ and let $s \triangleq \mathsf{lst}(\rho) \in \mathrm{St}_{\mathcal{G}_U}$ and $s' \triangleq \mathsf{unw}(s) = \mathsf{lst}(\mathsf{unw}_{trk}(\rho)) \in \mathrm{St}_{\mathcal{G}}$. Then, by the semantics definition, we have that there exists an s-total strategy $\mathsf{f}_A \in \mathrm{Str}_{\mathcal{G}_U}(A, s)$ such that, for all plays $\pi \in \mathrm{Play}_{\mathcal{G}_U}(\mathsf{f}_A)$, it holds that $\mathcal{G}_U, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$. Moreover, by the inductive hypothesis, it holds that $\mathcal{G}, \mathsf{unw}_{pth}(\rho \cdot \pi_{\geq 1}), 0, \mathsf{unw}_{trk}(\rho) \models \psi$. To prove the statement, it is left to show that there exists an s'-total strategy $\mathsf{f}'_A \in \mathrm{Str}_{\mathcal{G}}(A, s')$ such that, for all plays $\pi' \in \mathrm{Play}_{\mathcal{G}}(\mathsf{f}'_A)$, there exists a play $\pi \in \mathrm{Play}_{\mathcal{G}_U}(\mathsf{f}_A)$ such that $\mathsf{unw}_{trk}(\rho) \cdot \pi'_{\geq 1} = \mathsf{unw}_{pth}(\rho \cdot \pi_{\geq 1})$. To do this, we first define an auxiliary function $\mathsf{h} : \mathrm{Trk}_{\mathcal{G}}(s') \rightharpoonup \mathrm{Trk}_{\mathcal{G}_U}(s)$ mapping back tracks of $\mathcal{G}$ into corresponding tracks of $\mathcal{G}_U$. This function, can be inductively defined by means of the following recursive properties:
  1. $s' \in \mathsf{dom}(\mathsf{h})$ and $\mathsf{h}(s') \triangleq s$;
  2. for all $\rho' \in \mathsf{dom}(\mathsf{h})$ and counterdecision $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag} \setminus A}$, it holds that $\rho' \cdot t' \in \mathsf{dom}(\mathsf{h})$ and $\mathsf{h}(\rho' \cdot t') \triangleq \mathsf{h}(\rho') \cdot t$, where $t' \triangleq \tau_{\mathcal{G}}(\mathsf{lst}(\rho'), \mathsf{d})$, $t \triangleq \tau_{\mathcal{G}_U}(\mathsf{lst}(\mathsf{h}(\rho')), \mathsf{d})$, and $\mathsf{d} \triangleq (\mathsf{f}_A(\mathsf{h}(\rho')), \mathsf{d}^c_A)$.

Finally, the strategy $\mathsf{f}'_A \in \mathrm{Str}_{\mathcal{G}}(A, s')$ is defined as follows: $\mathsf{f}'_A(\rho') \triangleq \mathsf{f}_A(\mathsf{h}(\rho'))$, for all $\rho' \in \mathsf{dom}(\mathsf{f}'_A) \triangleq \mathsf{dom}(\mathsf{h})$. By a simple induction on the length of the play π', we can prove that f'_A actually satisfies the required property. Hence, we obtain that if $\mathcal{G}_U, \rho, \rho_p \models \langle\langle \mathrm{A} \rangle\rangle \psi$ then $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models \langle\langle \mathrm{A} \rangle\rangle \psi$.

- ($\varphi = \langle\langle \mathrm{A} \rangle\rangle \psi$, $\Leftarrow$)
  Suppose that $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models \langle\langle \mathrm{A} \rangle\rangle \psi$ and let $s \triangleq \mathsf{lst}(\rho) \in \mathrm{St}_{\mathcal{G}_U}$ and $s' \triangleq \mathsf{unw}(s) = \mathsf{lst}(\mathsf{unw}_{trk}(\rho)) \in \mathrm{St}_{\mathcal{G}}$. Then, by the semantics definition, we have that there exists an s'-total strategy $\mathsf{f}'_A \in \mathrm{Str}_{\mathcal{G}}(A, s')$ such that, for all plays $\pi' \in \mathrm{Play}_{\mathcal{G}}(\mathsf{f}'_A)$, it holds that $\mathcal{G}, \mathsf{unw}_{trk}(\rho) \cdot \pi'_{\geq 1}, 0, \mathsf{unw}_{trk}(\rho) \models \psi$. Let $\mathsf{f}_A \in \mathrm{Str}_{\mathcal{G}_U}(A, s)$ be the strategy defined as follows: $\mathsf{f}_A(\rho) \triangleq \mathsf{f}'_A(\mathsf{unw}_{trk}(\rho))$, for all $\rho \in \mathrm{Trk}_{\mathcal{G}_U}(s)$. It is easy to see that, for all plays $\pi \in \mathrm{Play}_{\mathcal{G}_U}(\mathsf{f}_A)$, it holds that $\mathsf{unw}_{pth}(\pi) \in \mathrm{Play}_{\mathcal{G}}(\mathsf{f}'_A)$. Thus, $\mathcal{G}, \mathsf{unw}_{trk}(\rho) \cdot \mathsf{unw}_{pth}(\pi)_{\geq 1}, 0, \mathsf{unw}_{trk}(\rho) \models \psi$, *i.e.*, $\mathcal{G}, \mathsf{unw}_{pth}(\rho \cdot \pi_{\geq 1}), 0, \mathsf{unw}_{trk}(\rho) \models \psi$. By the inductive hypothesis, it holds that $\mathcal{G}_U, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$. Therefore, we obtain that if $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models \langle\langle \mathrm{A} \rangle\rangle \psi$ then $\mathcal{G}_U, \rho, \rho_p \models \langle\langle \mathrm{A} \rangle\rangle \psi$. ∎

As an immediate corollary, we obtain that RATL* also enjoys the tree model property.

COROLLARY 3.4 (RATL* Tree Model Property)
RATL* enjoys the tree model property.

PROOF. Consider a formula φ and suppose that it is satisfiable. Then, there is a CGS $\mathcal{G}$ such that $\mathcal{G} \models \varphi$. By Theorem 3.3, φ is satisfied at the root of the unwinding $\mathcal{G}_U$ of $\mathcal{G}$. Thus, since $\mathcal{G}_U$ is a CGT, we immediately have that φ is satisfied on a tree model. ∎

4 Expressiveness and Succinctness

In this section, we compare RATL* with other logics derived from it. The basic comparisons are in terms of *expressiveness* and *succinctness*.

Let $\mathtt{L}_1$ and $\mathtt{L}_2$ be two logics whose semantics are defined on the same kind of structure. We say that $\mathtt{L}_1$ is *as expressive* $\mathtt{L}_2$ iff every formula in $\mathtt{L}_2$ is logically equivalent to some formula in $\mathtt{L}_1$. If $\mathtt{L}_1$ is as expressive as $\mathtt{L}_2$, but there is a formula in $\mathtt{L}_1$ that is not logically equivalent to any formula in $\mathtt{L}_2$, then $\mathtt{L}_1$ is *more expressive* than $\mathtt{L}_2$. If $\mathtt{L}_1$ is as expressive as $\mathtt{L}_2$ and vice versa, then $\mathtt{L}_1$ and $\mathtt{L}_2$ are *expressively equivalent*. Note that, in the case $\mathtt{L}_1$ is more expressive than $\mathtt{L}_2$, there are two sets of structures $\mathscr{M}_1$ and $\mathscr{M}_2$ and an $\mathtt{L}_1$ formula φ such that, for all $\mathcal{M}_1 \in \mathscr{M}_1$ and $\mathcal{M}_2 \in \mathscr{M}_2$, it holds that $\mathcal{M}_1 \models \varphi$ and $\mathcal{M}_2 \not\models \varphi$ and, for all $\mathtt{L}_2$ formulas φ', it holds that there are two models $M_1 \in \mathscr{M}_1$ and $\mathcal{M}_2 \in \mathscr{M}_2$ such that $\mathcal{M}_1 \models \varphi'$ iff $\mathcal{M}_2 \models \varphi'$. Intuitively, each $\mathtt{L}_2$ formula is not able to distinguish between two models that instead are different with respect to $\mathtt{L}_1$.

We define now the comparison of the two logics $\mathtt{L}_1$ and $\mathtt{L}_2$ in terms of succinctness, which measures the necessary blow-up when translating between them. Note that comparing logics in terms of succinctness makes sense also when the logics are not expressively equivalent, by focusing on their common fragment. In fact, a logic $\mathtt{L}_1$ can be more expressive than a logic $\mathtt{L}_2$, but at the same time, less succinct than the latter. Formally, we say that $\mathtt{L}_1$ is (at least) *exponentially more succinct* than $\mathtt{L}_2$ iff there exist two infinite lists of models $\{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$ and of $\mathtt{L}_1$ formulas $\{\varphi_1, \varphi_2, \ldots\}$, with $\mathcal{M}_i \models \varphi_i$ and $\mathsf{lng}(\varphi_i) = \mathtt{O}(\mathsf{p}_1(i))$, where $\mathsf{p}_1(n)$ is a polynomial, *i.e.*, $\mathsf{lng}(\varphi_i)$ is polynomial in $i \in \mathbb{N}$, such that, for all $\mathtt{L}_2$ formulas φ, it holds that if $\mathcal{M}_i \models \varphi$ then $\mathsf{lng}(\varphi) \geq 2^{\mathsf{p}_2(i)}$, where $\mathsf{p}_2(n)$ is another polynomial, *i.e.*, $\mathsf{lng}(\varphi)$ is (at least) exponential in i.

We now discuss expressiveness and succinctness of RATL* with respect to ATL* as well as some extensions/restrictions of RATL*. In particular, we consider the logics RPATL* and PATL* to be, respectively, RATL* and ATL* augmented with the past-time operators *"previous"* and *"since"*, which dualize the future-time operators *"next"* and *"until"* as in pLTL [28] and pCTL* [21, 22] (see Appendix B, for the full definition). Note that PATL* still contains the present proposition and that, as for pCTL*, the semantics of its quantifiers is as for ATL*, where the past is considered linear, *i.e.*, deterministic. Moreover, we consider the logics R$^-$ATL*, P$^-$ATL*, and RP$^-$ATL* to be, respectively, the syntactical restriction of RATL*, PATL*, and RPATL* in which the use of the distinguished atomic proposition present is not allowed. On one hand, we have that all mentioned logics, but R$^-$ATL* and RP$^-$ATL*, are expressively equivalent. On the other hand, the ability to refer to the past makes all of them at least exponentially more succinct than the corresponding ones without the past. For example, a PATL* formula φ can be translated into an equivalent ATL* one $\varphi^\star$, but $\varphi^\star$ may require a non-elementary space in $\mathsf{lng}(\varphi)$ (shortly, we say that PATL* is non-elementary reducible to ATL*). Note that, to get a better complexity for this translation is not an easy question. Indeed, it would improve the non-elementary reduction from *first order logic* to LTL, which is an outstanding open problem [17]. All the discussed results are reported in the following theorem. Observe that, to make clear the semantics we are using in the different points of the proof, we distinguish between the classic and the relentful ones by means of the modeling relations $\models_C$ and $\models_R$, respectively.

THEOREM 4.1 (Reductions)
The following properties hold:

1. ATL* (resp., PATL*) is linearly reducible to RATL* (resp., RPATL*);
2. RPATL* (resp., RP$^-$ATL*) is linearly reducible to PATL* (resp., P$^-$ATL*);
3. RPATL* (resp., RP$^-$ATL*) is non-elementarily reducible to RATL* (resp., R$^-$ATL*);
4. PATL* is non-elementarily reducible to ATL*;
5. R$^-$ATL* and P$^-$ATL* are at least exponentially more succinct than ATL*;
6. R$^-$ATL* is less expressive than ATL*.

PROOF. To prove Items 1-4, we describe a recursive translation from the input logics to the output ones, defined by structural induction on the formulas. For Items 5-6, instead, we directly exhibit a family of specifications expressed in the formalisms under analysis, which witnesses the asserted results.

Let φ and $\varphi^\star$ be, respectively, the input and output formulas for the first four statements. Moreover, consider the recursive syntactic modification $\overline{\cdot} : \text{RPATL}^* \to \text{RPATL}^*$ defined in the following, where the internal function $\mathsf{h} : 2^{\text{Ag}} \times \text{RPATL}^* \to \text{RPATL}^*$ has to be suitably specified in dependence of the item of interest:

- $\overline{\mathsf{present}} \triangleq \mathsf{present}$;
- $\overline{p} \triangleq p$, for $p \in \text{AP}$;
- $\overline{\mathsf{Op}\,\varphi} \triangleq \mathsf{Op}\,\overline{\varphi}$, for $\mathsf{Op} \in \{\neg, \mathsf{X}, \mathsf{Y}, \tilde{\mathsf{Y}}\}$;
- $\overline{\varphi_1 \mathsf{Op}\,\varphi_2} \triangleq \overline{\varphi_1}\mathsf{Op}\overline{\varphi_2}$, for $\mathsf{Op} \in \{\wedge, \vee, \mathsf{U}, \mathsf{S}, \mathsf{R}, \mathsf{B}\}$;
- $\overline{[\![A]\!]\psi} \triangleq \overline{\neg\langle\!\langle A\rangle\!\rangle\neg\psi}$;
- $\overline{\langle\!\langle A\rangle\!\rangle\psi} \triangleq \mathsf{h}(A, \overline{\psi})$.

Intuitively, the function $\overline{\cdot}$ simply replaces each occurrence of a strategic modality $\langle\langle A\rangle\rangle$ or $[\![A]\!]$ of the original logic with an opportune equivalent formula of the resulting logic. In particular, it does not introduce any additional present proposition or past-time operator, if the rewriting done by h does the same. Furthermore, in case $|\mathsf{h}(A,\psi)| = \mathtt{O}(|\psi|)$, we have that $|\overline{\varphi}| = \mathtt{O}(|\varphi|)$.

By means of a standard inductive proof, one can easily show the first two results.

- Item 1: $\mathcal{G} \models_C \varphi$ iff $\mathcal{G} \models_R \varphi^\star$, where $\varphi^\star \triangleq \overline{\varphi}$ with $\mathsf{h}(A,\psi) \triangleq \langle\langle A\rangle\rangle \mathsf{F}\,(\mathsf{present} \wedge \psi)$. Intuitively, we exploit the ability of RATL* (resp., RPATL*) to determine the present moment in order to simulate the classic interpretation of strategic modalities, in which the LTL (resp., pLTL) formula[2] ψ has to be verified from the current moment on.
- Item 2: $\mathcal{G} \models_R \varphi$ iff $\mathcal{G} \models_C \varphi^\star$, where $\varphi^\star \triangleq \overline{\varphi}$ with $\mathsf{h}(A,\psi) \triangleq \langle\langle A\rangle\rangle \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge \psi)$, in which $\mathsf{P}\,\psi'$ is the corresponding past-time operator for $\mathsf{F}\,\psi'$ and $\tilde{\mathsf{Y}}\,\psi'$ is the weak previous time operator that is true if either ψ' is true in the previous time-step or such a time-step does not exist. Intuitively, we ensure that the temporal property ψ is verified from the origin of the time, by coming back to this instant, from each possible moment, via the past-time operator P and the formula $\tilde{\mathsf{Y}}\,\mathfrak{f}$ that is true only at the initial state of a model.

Before continuing with the following two items, we need to introduce the auxiliary function $\widetilde{\cdot} : \text{RPATL}^* \to \text{RATL}^*$ defined as follows, which translates each formula of a suitable *separated normal form*[3] into an equivalent one without past-time operators. Note that the particular kind of normal form later on specified depends on the semantics of the fragment under analysis.

- $\widetilde{\mathsf{present}} \triangleq \mathsf{present}$;
- $\widetilde{p} \triangleq p$, for $p \in \text{AP}$;
- $\widetilde{\mathsf{Op}\,\psi} \triangleq \mathsf{Op}\,\widetilde{\psi}$, for $\mathsf{Op} \in \{\neg, \mathsf{X}, \langle\langle A\rangle\rangle, [\![A]\!]\}$;
- $\widetilde{\psi_1 \mathsf{Op}\,\psi_2} \triangleq \widetilde{\psi_1}\,\mathsf{Op}\,\widetilde{\psi_2}$, for $\mathsf{Op} \in \{\wedge, \vee, \mathsf{U}, \mathsf{R}\}$;
- $\widetilde{\mathsf{Y}\,\psi} \triangleq \mathfrak{f}$;
- $\widetilde{\tilde{\mathsf{Y}}\,\psi} \triangleq \mathfrak{t}$;
- $\widetilde{\psi_1 \mathsf{Op}\,\psi_2} \triangleq \widetilde{\psi_2}$, for $\mathsf{Op} \in \{\mathsf{S}, \mathsf{B}\}$.

Intuitively, the function $\widetilde{\cdot}$ eliminates all past-time operators by assuming that, due to the specific semantics of the fragment, they are only evaluated at the starting moment. Thus, no previous time-step exists. Therefore, Y and $\tilde{\mathsf{Y}}$ are directly solved by replacing them with the opportune truth value. The operators S and B, instead, to be verified at the beginning of the model, necessarily require that their second argument is immediately checked.

Now, it only remains to introduce the two separated normal forms used to prove Items 3 and 4. Informally, for the former, starting from each possible strategic modality, we need that a past-time operator is reached without passing through a future-time one. For the latter, in addition to the previous constraint, we also require that there are no strategic modalities containing a past-time operator. Formally, the followings hold.

- Item 3: $\mathcal{G} \models_R \varphi$ iff $\mathcal{G} \models_R \varphi^\star$, where $\varphi^\star \triangleq \widetilde{\overline{\varphi}}$ with $\mathsf{h}(A,\psi) \triangleq \langle\langle A\rangle\rangle \bigvee_{i\in I} \psi_i^{ps} \wedge \psi_i^{pr} \wedge \psi_i^{ft}$, in which $\bigvee_{i\in I} \psi_i^{ps} \wedge \psi_i^{pr} \wedge \psi_i^{ft}$ is the pLTL formula in separated normal form obtained by applying to ψ the well-known Separation Theorem (see Theorem 2.4 of [17]). Observe that

[2] Actually, ψ might not be an LTL (resp., pLTL) formula, but it can be seen as such by considering the internal state formulas as fresh atomic propositions.

[3] A pLTL formula is in *separated normal form* [17] if no past-time (resp., future-time) operator is in the scope of a future time (resp., past-time) one.

ψ_i^{ps}, ψ_i^{pr}, and ψ_i^{ft} are, respectively, pure-past, pure-present, and pure-future formulas[4]. Intuitively, we first ensure that, in each strategic modality, all past-time operators are not in the scope of a future-time one by means of the function $\bar{\cdot}$. Then, since the related temporal formulas are checked from the beginning of the model, we simply eliminate all past-time operators via the function $\widetilde{\cdot}$. Note that the non-elementary blow-up is due to the size of the set of indexes I, which is inherited from the use of the Separation Theorem.

- The proof of Item 4 proceeds similarly to the translation of pCTL* into CTL* (see Lemma 3.3 and Theorem 3.4 of [21, 22]). However, here we have two important differences. First, when we apply the Separation Theorem to ψ in order to obtain the equivalent path formula $\bigvee_{i\in I} \psi_i^{ps} \wedge \psi_i^{pr} \wedge \psi_i^{ft}$ in separated normal form, we need to evaluate the semantics of all the occurrences of the present proposition by substituting them with $\mathfrak{f}$, in the pure-past and pure-future formulas ψ_i^{ps} and ψ_i^{ft}, and with $\mathfrak{t}$ in the pure-present ones ψ_i^{pr}. In particular, *w.l.o.g.*, we require that each of the occurrences in ψ_i^{ps} and ψ_i^{ft} is, respectively, in the scope of a previous (Y or $\tilde{\mathsf{Y}}$) or next (X) time operator. Then, since the strategic modality $\langle\langle A\rangle\rangle$ cannot be simply seen as existential due to the internal universal quantifications over paths, we cannot push it directly in front of the pure-future formulas by commuting it with the disjunction $\bigvee_{i\in I}$. Instead, we have to suitably determine which of the pure-past formulas are verified and then push the modality in front of the disjunction of the corresponding pure-present and pure-future formulas. Formally, we have that $\mathcal{G} \models_C \varphi$ iff $\mathcal{G} \models_C \varphi^\star$, where $\varphi^\star \triangleq \widetilde{\overline{\varphi}}$ with $\mathsf{h}(A,\psi) \triangleq \bigvee_{I'\subseteq I}^{I'\neq\emptyset} \left((\bigwedge_{i\in I'} \psi_i^{ps\prime}) \wedge \langle\langle A\rangle\rangle \bigvee_{i\in I'} \psi_i^{pr\prime} \wedge \psi_i^{ft\prime} \right)$, in which $\psi_i^{ps\prime}$, $\psi_i^{pr\prime}$, and $\psi_i^{ft\prime}$ are, respectively, obtained from ψ_i^{ps}, ψ_i^{pr}, and ψ_i^{ft} by replacing the present proposition as described above. Intuitively, we first propagate out the past-time operators from every strategic modalities via the function $\bar{\cdot}$. Then, since they are immediately checked at the root state of the model, before every strategic reasoning is evaluated, we eliminate them by applying the function $\widetilde{\cdot}$. As for the previous item, the origin of the non-elementary blow-up resides in the Separation Theorem.

It only remains to prove the last two statements about succinctness and expressiveness.

- Item 5 is derived by applying the following reasoning. In [27], pLTL was proved to be exponentially more succinct than LTL, by showing that every formula $\psi_n \triangleq \mathsf{G}\,(\bigwedge_{i=1}^{n}(p_i \leftrightarrow \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f})\wedge p_i)) \rightarrow (p_0 \leftrightarrow \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f})\wedge p_0)))$ has only LTL equivalents with length $\Omega(2^n)$ (see Theorem 3.1). Now, since each ψ_n is equivalent to the pCTL* formula $\mathsf{A}\psi_n$, which in turn is equivalent to the P$^-$ATL* formula $\varphi_P \triangleq [\![\mathrm{Ag}]\!]\psi_n$, we immediately derive that P$^-$ATL* is at least exponentially more succinct than ATL*. To prove that the same holds for R$^-$ATL*, first observe that ψ_n ensures every state agreeing with the initial one on the truth value of $p_1,\ldots,p_n$ to also agree with it on p_0. The comparison between the different states is done in pLTL by referring to the root of the model via the past-time operator P and the formula $\tilde{\mathsf{Y}}\,\mathfrak{f}$. Now, by exploiting the observation at the base of Item 2 in the reverse order, we can simulate the comparison by means of the relentful semantics of R$^-$ATL*. Indeed, it is easy to show that the R$^-$ATL formula $\varphi_R \triangleq [\![\mathrm{Ag}]\!]\mathsf{G}\,(\bigwedge_{i=1}^{n}(p_i \leftrightarrow [\![\mathrm{Ag}]\!]p_i) \rightarrow (p_0 \leftrightarrow [\![\mathrm{Ag}]\!]p_0))$ is equivalent to φ_P, *i.e.*, $\mathcal{G} \models_R \varphi_R$ iff $\mathcal{G} \models_C \varphi_P$. Thus, the result holds for R$^-$ATL* too.
- Similarly to the previous item, the proof of Item 6 follows by exploiting a related result for a fragment of R$^-$ATL*. Indeed, in [25], it was proved that the CTL formula $\mathsf{EF}\,(\mathsf{EX}\,p \wedge \mathsf{EX}\,\neg p)$ has no m$^-$CTL* equivalent (see Theorem 3.4). Now, consider a model in which there is

[4] A pure-past (resp., pure-future) formula contains only past-time (resp., future-time) operators. In addition, pure-present formulas do not contain any temporal operator at all [17].

just one agent. Then, it is evident that every CTL (resp., m–CTL*) formula is equivalent to the ATL (resp., R–ATL*) one in which the two path quantifiers E and A are replaced by the strategic modalities $\langle\langle \mathrm{Ag} \rangle\rangle$ and $[\![\mathrm{Ag}]\!]$, respectively. Consequently, the ATL formula $\langle\langle \mathrm{Ag} \rangle\rangle \mathsf{F}\,((\langle\langle \mathrm{Ag} \rangle\rangle \mathsf{X}\, p) \wedge (\langle\langle \mathrm{Ag} \rangle\rangle \mathsf{X}\, \neg p))$ has no R–ATL* equivalent one. ∎

As an immediate consequence of some combinations of the results shown into the previous theorem, it is easy to prove the following corollary.

COROLLARY 4.2 (Expressiveness)
RATL*, PATL*, P–ATL*, and RPATL* have the same expressive power of ATL*. RP–ATL* has the same expressive power of R–ATL*, and both are less expressive than ATL*. Moreover, all of them are at least exponentially more succinct than ATL*.

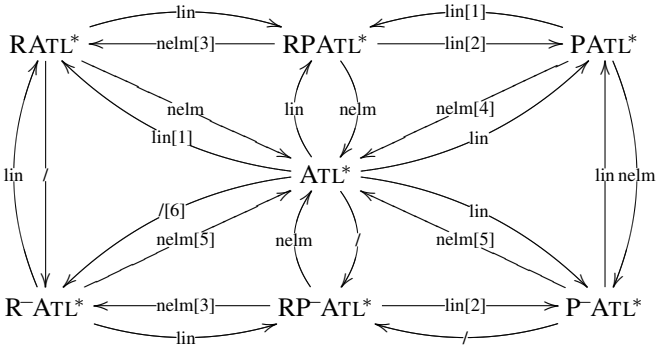

FIG. 4: Expressive power and succinctness hierarchy.

Figure 4 summarizes all the above results regarding expressiveness and succinctness. The acronym *"lin"* (resp., *"nelm"*) means that the translation exists and it is linear (resp., non-elementarily) in the size of the formula, and *"/"* means that such a translation is impossible. The numbers in brackets represent the item of Theorem 4.1 in which the translation is shown. We use no numbers when the translation is trivial or comes by a composition of existing ones.

We conclude this section by describing an example of reduction from RATL* and PATL* to ATL*. Consider the RATL* formula

$$\varphi_R \triangleq [\![\mathrm{Ag}]\!]\mathsf{G}\,(p \vee \langle\langle A \rangle\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2)),$$

which asserts that, in every state falsifying the condition p, the team of agents A has the ability to enforce the play started at the initial state to reach first p_1 and then p_2. Observe that, due to the relentful semantics, it does not matter if these two events happen before, in between, or after the present moment. However, their relative order has to be respected. Before continuing, one can easily note a similarity with the specification of the planning example previously described, in which the atomic proposition p is in place of the formulas η_i^1 and η_i^2, while $\langle\langle A \rangle\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2)$ stands for η_i^3. By means of the construction of Item 2 of Theorem 4.1, we immediately obtain the equivalent PATL* formula

$$\varphi_P \triangleq [\![\mathrm{Ag}]\!]\mathsf{G}\,(p \vee \langle\langle A \rangle\rangle \mathsf{P}\,((\tilde{\mathsf{Y}}\, \mathfrak{f}) \wedge \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2))),$$

in which the relentful semantics is simulated via the past-time operators (actually, this is a simplification of what it is obtained by the construction, since we avoided to modify the external part $[\![\mathrm{Ag}]\!]\mathsf{G}$ that is only verified at the initial state). At this point, by applying the

Separation Theorem to $\mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2))$, we obtain the equivalent formula in separated normal form $\mathsf{P}\,(p_2 \wedge \mathsf{P}\,p_1) \vee (\mathsf{P}\,p_1 \wedge \mathsf{F}\,p_2) \vee \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2)$. Consequently, via the function $\bar{\cdot}$ defined in Item 4 of Theorem 4.1, after few simplifications, we have that the formula

$$\langle\!\langle A \rangle\!\rangle \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2))$$

is translated to

$$\mathsf{P}\,(p_2 \wedge \mathsf{P}\,p_1) \vee (\mathsf{P}\,p_1 \wedge \langle\!\langle A \rangle\!\rangle \mathsf{F}\,p_2) \vee \langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2).$$

Thus, $\varphi_P \equiv [\![\mathrm{Ag}]\!]\psi$, where

$$\psi = \mathsf{G}\,(p \vee \mathsf{P}\,(p_2 \wedge \mathsf{P}\,p_1) \vee (\mathsf{P}\,p_1 \wedge \langle\!\langle A \rangle\!\rangle \mathsf{F}\,p_2) \vee \langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2)).$$

Now, we can apply the same translation function to the external modality $[\![\mathrm{Ag}]\!]\psi$, which is converted into $\neg\langle\!\langle \mathrm{Ag} \rangle\!\rangle \neg\psi$. First, note that

$$\begin{aligned}\neg\psi &\equiv \mathsf{F}\,(\neg p \wedge \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1) \wedge (\mathsf{H}\,\neg p_1 \vee \neg\langle\!\langle A \rangle\!\rangle \mathsf{F}\,p_2) \wedge \neg\langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2)) \\ &= \mathsf{F}\,(\xi \wedge \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1) \wedge (\mathsf{H}\,\neg p_1 \vee \zeta)),\end{aligned}$$

where $\xi = \neg p \wedge \neg\langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\,p_2)$ and $\zeta = \neg\langle\!\langle A \rangle\!\rangle \mathsf{F}\,p_2$. So, by putting the internal part in disjunctive normal form and then simplifying it, we obtain that

$$\neg\psi \equiv \mathsf{F}\,(\xi \wedge \mathsf{H}\,\neg p_1) \vee \mathsf{F}\,(\eta \wedge \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1)),$$

where $\eta = \neg p \wedge \zeta$. By manipulating the two disjuncts through the Separation Theorem, we have that

$$\mathsf{F}\,(\xi \wedge \mathsf{H}\,\neg p_1) \equiv (\mathsf{H}\,\neg p_1) \wedge (\neg p_1)\mathsf{U}\,(\xi \wedge \neg p_1)$$

and

$$\begin{aligned}\mathsf{F}\,(\eta \wedge \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1)) \equiv\ & (\mathsf{H}\,\neg p_1) \wedge (\neg p_1)\mathsf{U}\,(\eta \wedge \neg p_1) \vee \\ & \vee (\mathsf{H}\,\neg p_1) \wedge (\neg p_1)\mathsf{U}\,(\neg p_2 \mathsf{U}\,(\eta \wedge \neg p_2)) \vee \\ & \vee \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1) \wedge (\neg p_2 \mathsf{U}\,(\eta \wedge \neg p_2)).\end{aligned}$$

So, by summing up the two obtained results and eliminating the first term of the second equivalence that results to be redundant, we immediately derive

$$\begin{aligned}\neg\psi \equiv\ & (\mathsf{H}\,\neg p_1) \wedge (\neg p_1)\mathsf{U}\,(\xi \wedge \neg p_1) \vee \\ & \vee (\mathsf{H}\,\neg p_1) \wedge (\neg p_1)\mathsf{U}\,(\neg p_2 \mathsf{U}\,(\eta \wedge \neg p_2)) \vee \\ & \vee \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1) \wedge (\neg p_2 \mathsf{U}\,(\eta \wedge \neg p_2)).\end{aligned}$$

By applying again the function $\bar{\cdot}$ defined in Item 4, after few simplifications, we have that the formula $\varphi_P \equiv \neg\langle\!\langle \mathrm{Ag} \rangle\!\rangle \neg\psi$ is translated to $\neg((\mathsf{H}\,\neg p_1) \wedge \langle\!\langle \mathrm{Ag} \rangle\!\rangle (\psi_1 \vee \psi_2) \vee \mathsf{H}\,(\neg p_2 \vee \mathsf{H}\,\neg p_1) \wedge \langle\!\langle \mathrm{Ag} \rangle\!\rangle \psi_3)$, where

$$\begin{aligned}\psi_1 &= (\neg p_1)\mathsf{U}\,(\xi \wedge \neg p_1), \\ \psi_2 &= (\neg p_1)\mathsf{U}\,(\neg p_2 \mathsf{U}\,(\eta \wedge \neg p_2)), \\ \psi_3 &= (\neg p_2)\mathsf{U}\,(\eta \wedge \neg p_2).\end{aligned}$$

Finally, by eliminating the past-time operators trough the function $\widetilde{\cdot}$, we obtain that the PATL* formula φ_P is definitively translated to

$$\begin{aligned}\varphi_C &= \widetilde{\overline{\varphi_P}}\\ &\equiv \neg((\neg p_1) \wedge \langle\!\langle \mathrm{Ag} \rangle\!\rangle (\psi_1 \vee \psi_2) \vee (\neg p_2) \wedge \langle\!\langle \mathrm{Ag} \rangle\!\rangle \psi_3)\\ &\equiv (p_1 \vee [\![\mathrm{Ag}]\!](\psi_1' \wedge \psi_2')) \wedge (p_2 \vee [\![\mathrm{Ag}]\!]\psi_3'),\end{aligned}$$

where

$$\begin{aligned}\psi_1' &= p_1 \mathsf{R}\, (\xi' \vee p_1),\\ \psi_2' &= p_1 \mathsf{R}\, (p_2 \mathsf{R}\, (\eta' \vee p_2)),\\ \psi_3' &= p_2 \mathsf{R}\, (\eta' \vee p_2),\\ \xi' &= p \vee \langle\!\langle A \rangle\!\rangle \mathsf{F}\, (p_1 \wedge \mathsf{F}\, p_2),\\ \eta' &= p \vee \langle\!\langle A \rangle\!\rangle \mathsf{F}\, p_2,\end{aligned}$$

which in turn is equivalent to

$$p_1 \wedge p_2 \vee p_1 \wedge [\![\mathrm{Ag}]\!]\psi_3' \vee [\![\mathrm{Ag}]\!](\psi_1' \wedge \psi_2').$$

After this tour in the translation process of a RATL* formula into an ATL* equivalent one, it is quite clear that there are properties whose description in the former logic is very easy, but that in the latter one becomes quite intricate and difficult to understand. However, the problem with ATL* does not reside only in the difficulty of writing and understanding of some formulas, but also in their succinctness. Consider, for example, the RATL* formula

$$[\![\mathrm{Ag}]\!]\mathsf{G}\, (p \vee \langle\!\langle A \rangle\!\rangle ((\mathsf{F}\, p_1) \wedge (\mathsf{F}\, p_2)))$$

and its PATL* equivalent

$$[\![\mathrm{Ag}]\!]\mathsf{G}\, (p \vee \langle\!\langle A \rangle\!\rangle \mathsf{P}\, ((\tilde{\mathsf{Y}}\, \mathfrak{f}) \wedge (\mathsf{F}\, p_1) \wedge (\mathsf{F}\, p_2))),$$

which, differently from the previous case, do not require a predetermined order between the two events p_1 and p_2. Now, by applying the Separation Theorem to the internal part $\mathsf{P}\, ((\tilde{\mathsf{Y}}\, \mathfrak{f}) \wedge (\mathsf{F}\, p_1) \wedge (\mathsf{F}\, p_2))$, we obtain the following formula in separated normal form

$$((\mathsf{P}\, p_1) \wedge (\mathsf{P}\, p_2)) \vee (\mathsf{P}\, p_1 \wedge \mathsf{F}\, p_2) \vee (\mathsf{P}\, p_2 \wedge \mathsf{F}\, p_1) \vee ((\mathsf{F}\, p_1) \wedge (\mathsf{F}\, p_2)),$$

which describes all four possible ways to split the two events in between the present moment. More in general, in case of $n \in \mathbb{N}$ events, the application of the theorem to the RATL* formula

$$[\![\mathrm{Ag}]\!]\mathsf{G}\, (p \vee \langle\!\langle A \rangle\!\rangle \bigwedge_{i=1}^{n} \mathsf{F}\, q_i)$$

would produce the PLTL formula

$$\bigvee_{I \subseteq [1,n]} (\bigwedge_{i \in I} \mathsf{P}\, q_i) \wedge (\bigwedge_{i \in [1,n] \setminus I} \mathsf{F}\, q_i),$$

whose length is exponential in n. Therefore, the ATL* equivalent would be exponential as well.

5 Alternating Tree Automata

In this section, we briefly introduce an automaton model used to solve efficiently the satisfiability and model-checking problems for RPATL*, by reducing them, respectively, to the emptiness and membership problems of the automaton. We recall that, in general, such an approach is only possible once the logic satisfies the invariance under unwinding. In fact, this property holds for RPATL*, as it is stated in Theorem 3.3.

5.1 Classic automata

Alternating tree automata [36] are a generalization of nondeterministic tree automata. Intuitively, while a nondeterministic automaton that visits a node of the input tree sends exactly one copy of itself to each of the successors of the node, an alternating automaton can send several copies of itself to the same successor. *Symmetric automata* [20] are a variation of classical (asymmetric) alternating automata in which it is not necessary to specify the direction (*i.e.*, the choice of the successors) of the tree on which a copy is sent. In fact, through two generalized directions (existential and universal moves), it is possible to send a copy of the automaton, starting from a node of the input tree, to one or all its successors. Hence, the automaton does not distinguish between directions. As a generalization of symmetric alternating automata, here we consider automata that can send copies to successor nodes, according to some entity choice. These automata are a slight variation of *automata over concurrent game structures* introduced in [43].

We now give the formal definition of symmetric and asymmetric alternating tree automata.

DEFINITION 5.1 (Symmetric Alternating Tree Automata)
A *symmetric alternating tree automaton* (*S*ATA, for short) is a tuple $\mathcal{A} \triangleq \langle \Sigma, \mathrm{E}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$, where Σ, E, and Q are non-empty finite sets of *input symbols*, *entities*, and *states*, respectively, $q_0 \in \mathrm{Q}$ is an *initial state*, F is an *acceptance condition* to be defined later, and $\delta : \mathrm{Q} \times \Sigma \to \mathtt{B}^+(\mathrm{D} \times \mathrm{Q})$ is an *alternating transition function*, where $\mathrm{D} = \{\Diamond, \Box\} \times 2^{\mathrm{E}}$ is an extended set of *abstract directions*, which maps each pair of states and input symbols to a positive Boolean combination on the set of propositions, *a.k.a. abstract moves*, of the following form: *existential* $((\Diamond, A), q)$ and *universal* $((\Box, A), q)$ propositions, with $A \subseteq \mathrm{E}$ and $q \in \mathrm{Q}$.

DEFINITION 5.2 (Asymmetric Alternating Tree Automata)
An *asymmetric alternating tree automaton* (*A*ATA, for short) is a tuple $\mathcal{A} \triangleq \langle \Sigma, \Delta, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$, where Σ, Q, q_0, and F are defined as for the symmetric one, Δ is a non-empty finite set of *real directions*, and $\delta : \mathrm{Q} \times \Sigma \to \mathtt{B}^+(\Delta \times \mathrm{Q})$ is an *alternating transition function* that maps each pair of states and input symbols to a positive Boolean combination on the set of propositions of the form $(d, q) \in \Delta \times \mathrm{Q}$, *a.k.a. real moves*.

A *nondeterministic tree automaton* (NTA, for short) is a special AATA in which each conjunction in the transition function δ has exactly one move (d, q) associated with each direction d. In addition, a *universal tree automaton* (UTA, for short) is a special AATA in which all the Boolean combinations that appear in δ are only conjunctions of moves.

In the following, we simply write ATA when we indifferently refer to its symmetric or asymmetric version.

The semantics of ATAs is now given through the following related concepts of run.

DEFINITION 5.3 (SATA Run)
A *run* of a SATA $\mathcal{A} = \langle \Sigma, \mathrm{E}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ on a Σ-labeled B^{E}-tree $\mathcal{T} = \langle \mathrm{T}, \mathsf{v} \rangle$, for a given set B, is a $(\mathrm{Q} \times \mathrm{T})$-labeled $\mathbb{N}$-tree $\mathcal{R} \triangleq \langle \mathrm{R}, \mathsf{r} \rangle$ such that *(i)* $\mathsf{r}(\varepsilon) = (q_0, \varepsilon)$ and *(ii)* for all nodes $y \in \mathrm{R}$ with $\mathsf{r}(y) = (q, x)$, there is a set of abstract moves $\mathrm{S} \subseteq \mathrm{D} \times \mathrm{Q}$ with $\mathrm{S} \models \delta(q, \mathsf{v}(x))$ such that, for all $(z, q') \in \mathrm{S}$, it holds that:

- if $z = (\Diamond, A)$ then there exists a choice $d \in \mathrm{B}^A$ such that, for all counterchoices $d' \in \mathrm{B}^{\mathrm{E} \setminus A}$, it holds that $(q', x \cdot (d, d')) \in \mathsf{labsuc}(y)$;
- if $z = (\Box, A)$ then, for all choices $d \in \mathrm{B}^A$, there exists a counterchoice $d' \in \mathrm{B}^{\mathrm{E} \setminus A}$ such that $(q', x \cdot (d, d')) \in \mathsf{labsuc}(y)$;

where $(d, d') \in \mathrm{B}^{\mathrm{E}}$ denotes composition of d and d', *i.e.*, the function such that $(d, d')_{\restriction A} = d$ and $(d, d')_{\restriction (\mathrm{E} \setminus A)} = d'$ and $\mathsf{labsuc}(y) \triangleq \{\mathsf{r}(y \cdot j) : j \in \mathbb{N} \wedge y \cdot j \in \mathrm{R}\}$ is the set of labels of successors of the node y in the run $\mathcal{R}$.

DEFINITION 5.4 (AATA Run)
A *run* of an AATA $\mathcal{A} = \langle \Sigma, \Delta, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ on a Σ-labeled Δ-tree $\mathcal{T} = \langle \mathrm{T}, \mathsf{v} \rangle$ is a $(\mathrm{Q} \times \mathrm{T})$-labeled $\mathbb{N}$-tree $\mathcal{R} \triangleq \langle \mathrm{R}, \mathsf{r} \rangle$ such that *(i)* $\mathsf{r}(\varepsilon) = (q_0, \varepsilon)$ and *(ii)* for all nodes $y \in \mathrm{R}$ with $\mathsf{r}(y) = (q, x)$, there is a set of real moves $\mathrm{S} \subseteq \Delta \times \mathrm{Q}$ with $\mathrm{S} \models \delta(q, \mathsf{v}(x))$ such that, for all $(d, q') \in \mathrm{S}$, there is an index $j \in [0, |\mathrm{S}|[$ for which it holds that $y \cdot j \in \mathrm{R}$ and $\mathsf{r}(y \cdot j) = (q', x \cdot d)$.

In the following, we consider ATAs along with the *parity* $\mathrm{F} = (\mathrm{F}_1, \ldots, \mathrm{F}_k) \in (2^{\mathrm{Q}})^+$ with $\mathrm{F}_1 \subseteq \ldots \subseteq \mathrm{F}_k = \mathrm{Q}$ (APT, for short) acceptance condition (see [26], for more). The number k of sets in F is called the *index* of the automaton. We also use ATAs with the *Co-Büchi* acceptance condition $\mathrm{F} \subseteq \mathrm{Q}$ (ACT, for short) that are APTs of index 2 in which the set of final states is represented by F_1 only.

Let $\mathcal{R} = \langle \mathrm{R}, \mathsf{r} \rangle$ be a run of an ATA $\mathcal{A}$ on a tree $\mathcal{T} = \langle \mathrm{T}, \mathsf{v} \rangle$ and $\mathrm{R}' \subseteq \mathrm{R}$ one of its branches. Then, by $\mathsf{inf}(\mathrm{R}') \triangleq \{q \in \mathrm{Q} : |\{y \in \mathrm{R}' : \exists x \in \mathrm{T} \,.\, \mathsf{r}(y) = (q, x)\}| = \omega\}$ we denote the set of states that occur infinitely often as labeling of the nodes in the branch R'. We say that a branch R' of $\mathcal{T}$ satisfies the parity acceptance condition $\mathrm{F} = (\mathrm{F}_1, \ldots, \mathrm{F}_k)$ iff the least index $i \in [1, k]$ for which $\mathsf{inf}(\mathrm{R}') \cap \mathrm{F}_i \neq \emptyset$ is even.

We now define the concept of language accepted by an ATA.

DEFINITION 5.5 (ATA Acceptance)
A SATA $\mathcal{A} = \langle \Sigma, \mathrm{E}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ (resp., AATA $\mathcal{A} = \langle \Sigma, \Delta, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$) *accepts* a Σ-labeled B^{E}-tree (resp., Δ-tree) $\mathcal{T}$ iff there exists a run $\mathcal{R}$ of $\mathcal{A}$ on $\mathcal{T}$ such that all its infinite branches satisfy the acceptance condition F, where the concept of satisfaction is dependent from of the definition of F.

By $\mathrm{L}(\mathcal{A})$ we denote the language accepted by the ATA $\mathcal{A}$, *i.e.*, the set of trees $\mathcal{T}$ accepted by $\mathcal{A}$. Moreover, $\mathcal{A}$ is said to be *empty* if $\mathrm{L}(\mathcal{A}) = \emptyset$. The *emptiness problem* for $\mathcal{A}$ is to decide whether $\mathrm{L}(\mathcal{A}) = \emptyset$.

In the sequel, we show how to reduce, for equivalence, a SATA to an AATA when it is *a priori* known the structure of the trees of interest.

THEOREM 5.6 (SATA-AATA Reduction)
Let $\mathcal{A} = \langle \Sigma, \mathrm{E}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ be a SATA and B be a finite set. Then there is an AATA $\mathcal{A}' = \langle \Sigma, \mathrm{B}^{\mathrm{E}}, \mathrm{Q}, \delta', q_0, \mathrm{F} \rangle$ such that every Σ-labeled B^{E}-tree is accepted by $\mathcal{A}$ iff it is accepted by $\mathcal{A}'$.

PROOF. The transition function δ' of $\mathcal{A}'$ is obtained from the one of $\mathcal{A}$ by substituting each existential $((\Diamond, A), q')$ and universal $((\Box, A), q')$ move with the formulas $\bigvee_{d \in \mathrm{B}^A} \bigwedge_{d' \in \mathrm{B}^{\mathrm{E} \setminus A}} ((d,$

$d'), q')$ and $\bigwedge_{d \in \mathrm{B}^{A}} \bigvee_{d' \in \mathrm{B}^{\mathrm{E} \setminus A}} ((d, d'), q')$, respectively. Directly from the Definition 5.3 of SATA run, the thesis immediately follows. ∎

5.2 Automata with satellite

As a generalization of ATA, here we also consider *alternating tree automata with satellites* (ATAS, for short), in a similar way it has been done in [25]. The satellite is used to take a bounded memory of the evaluated part of a path in a given structure and it is kept apart from the main automaton as it allows to show a tight complexity for the decision problems. We use symmetric ATAS (SATAS, for short) for the solution of the satisfiability problem and asymmetric ATAS (AATAS, for short) for the model-checking one.

We now formally define this new type of automaton.

DEFINITION 5.7 (Alternating Tree Automata with Satellite)
A *symmetric* (resp., *asymmetric*) *alternating tree automaton with satellite* (*S*ATAS (resp., *A*ATAS), for short) is a tuple $\langle \mathcal{A}, \mathcal{S} \rangle$, where $\mathcal{A} \triangleq \langle \Sigma \times \mathrm{P}, \mathrm{E}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ (resp., $\mathcal{A} \triangleq \langle \Sigma \times \mathrm{P}, \Delta, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$) is a SATA (resp., AATA) and $\mathcal{S} \triangleq \langle \Sigma, \mathrm{P}, \zeta, p_0 \rangle$ is a *deterministic safety word automaton*[5], *a.k.a. satellite*, where P is a non-empty finite set of *states*, $p_0 \in \mathrm{P}$ is an *initial state*, and $\zeta : \mathrm{P} \times \Sigma \to \mathrm{P}$ is a *deterministic transition function* that maps a state and an input symbol to a state. The sets Σ and E (resp., Δ) are, respectively, the *alphabet* and the *entity set* (resp., *direction sets*) of the ATAS $\langle \mathcal{A}, \mathcal{S} \rangle$.

In the following definition, we formalize the concept of a language being accepted by an ATAS.

DEFINITION 5.8 (ATAS Acceptance)
A Σ-labeled B^{E}-tree (resp., Δ-tree) $\mathcal{T}$ is accepted by a SATAS (resp., AATAS) $\langle \mathcal{A}, \mathcal{S} \rangle$, where $\mathcal{A} \triangleq \langle \Sigma \times \mathrm{P}, \mathrm{E}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ (resp., $\mathcal{A} = \langle \Sigma \times \mathrm{P}, \Delta, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$) and $\mathcal{S} = \langle \Sigma, \mathrm{P}, \zeta, p_0 \rangle$, iff it is accepted by the product-automaton $\mathcal{A}^{\star} \triangleq \langle \Sigma, \mathrm{E}, \mathrm{Q} \times \mathrm{P}, \delta^{\star}, (q_0, p_0), \mathrm{F}^{\star} \rangle$ (resp., $\mathcal{A}^{\star} \triangleq \langle \Sigma, \Delta, \mathrm{Q} \times \mathrm{P}, \delta^{\star}, (q_0, p_0), \mathrm{F}^{\star} \rangle$) with $\delta^{\star}((q, p), \sigma) \triangleq \delta(q, (\sigma, p))[q' \in \mathrm{Q}/(q', \zeta(p, \sigma))]$, where by $f[x \in \mathrm{X}/y]$ we denote the formula in which all occurrences of x in f are replaced by y, and $\mathrm{F}^{\star}$ is the acceptance condition directly derived from F.

In words, $\delta^{\star}((q, p), \sigma)$ is obtained by substituting in $\delta(q, (\sigma, p))$ each occurrence of a state q' with a tuple of the form (q', p'), where $p' = \zeta(p, \sigma)$ is the new state of the satellite. By $\mathrm{L}(\langle \mathcal{A}, \mathcal{S} \rangle)$ we denote the language accepted by the ATAS $\langle \mathcal{A}, \mathcal{S} \rangle$.

In the following, we mainly consider ATAS along with the parity acceptance condition (APTS, for short), where $\mathrm{F}^{\star} \triangleq (\mathrm{F}_1 \times \mathrm{P}, \ldots, \mathrm{F}_k \times \mathrm{P})$.

Note that satellites are just a convenient way to describe an ATA in which the state space can be partitioned into two components, one of which is deterministic, independent from the other, and having no influence on the acceptance. Indeed, it is just a matter of technicality to see that automata with satellites inherit all the closure properties of alternating automata. In particular, we prove how to translate an AAPTS into an equivalent NPT with only an exponential blow-up in the number of the main states.

THEOREM 5.9 (AAPTS Nondeterminization)
Let $\langle \mathcal{A}, \mathcal{S} \rangle$ be an AAPTS, where the main automaton $\mathcal{A}$ has n states and index k and the satellite $\mathcal{S}$ has m states. Then, there is an NPT $\mathcal{N}^{\star}$ with $2^{\mathtt{O}((n \cdot k) \cdot \log(n \cdot k) + \log(m))}$ states and index $\mathtt{O}(n \cdot k)$, such that $\mathrm{L}(\mathcal{N}^{\star}) = \mathrm{L}(\langle \mathcal{A}, \mathcal{S} \rangle)$.

[5] A safety word automaton is an automaton without acceptance condition, *i.e.*, all words for which there exists a run are accepted.

PROOF. To deduce the thesis, we use the Muller-Schupp exponential-time nondeterminization procedure [37] that leads from the AAPT $\mathcal{A}$ to an NPT $\mathcal{N}$, with $2^{\mathtt{O}((n\cdot k)\cdot\log(n\cdot k))}$ states and index $\mathtt{O}(n\cdot k)$, such that $\mathrm{L}(\mathcal{N}) = \mathrm{L}(\mathcal{A})$. Since an NPT is a particular AAPT, we immediately have that $\mathrm{L}(\langle\mathcal{N},\mathcal{S}\rangle) = \mathrm{L}(\langle\mathcal{A},\mathcal{S}\rangle)$. By taking the product-automaton between $\mathcal{N}$ and the satellite $\mathcal{S}$, as described in Definition 5.8 of ATAS acceptance, we obtain a new NPT $\mathcal{N}^\star$, with $2^{\mathtt{O}((n\cdot k)\cdot\log(n\cdot k)+\log(m))}$ states and index $\mathtt{O}(n\cdot k)$, such that $\mathrm{L}(\mathcal{N}^\star) = \mathrm{L}(\langle\mathcal{N},\mathcal{S}\rangle)$. Hence, it is evident that $\mathrm{L}(\mathcal{N}^\star) = \mathrm{L}(\langle\mathcal{A},\mathcal{S}\rangle)$. ∎

The following theorem, directly derived by a proof idea of [25], shows how the separation between $\mathcal{A}$ and $\mathcal{S}$ gives a tight analysis of the complexity of the relative emptiness problem.

THEOREM 5.10 (APTS Emptiness)
The *emptiness problem* for an APTS $\langle\mathcal{A},\mathcal{S}\rangle$ with alphabet size h, where the main automaton $\mathcal{A}$ has n states and index k and the satellite $\mathcal{S}$ has m states, can be decided in time $2^{\mathtt{O}(\log(h)+(n\cdot k)\cdot((n\cdot k)\cdot\log(n\cdot k)+\log(m)))}$.

PROOF. The proof proceeds in two steps, the first of which is used only if $\mathcal{A}$ is a SATA, in order to translate it into an AATA. First, in order to obtain a linear translation from SATAs to AATAs, we use a bounded model theorem (see Theorem 2 of [43]), which asserts that a SATA $\mathcal{A}$ accepts a tree iff it accepts a $|\mathrm{Z}\times\mathrm{E}|^{|\mathrm{E}|}$-bounded tree, where Z is the set of abstract moves used in its transition function. By Theorem 5.6, there is an AATA $\mathcal{A}'$, with the same set of states and acceptance condition of the original automaton $\mathcal{A}$ and a set $\mathrm{Z}\times\mathrm{E}^{\mathrm{E}}$ of directions, such that $\mathrm{L}(\mathcal{A}') = \emptyset$ iff $\mathrm{L}(\mathcal{A}) = \emptyset$. Hence, by the definition of ATAS, we obtain that $\mathrm{L}(\langle\mathcal{A}',\mathcal{S}\rangle) = \emptyset$ iff $\mathrm{L}(\langle\mathcal{A},\mathcal{S}\rangle) = \emptyset$. Consequently, by Theorem 5.9, we obtain an NPT $\mathcal{N}^\star$, with $2^{\mathtt{O}((n\cdot k)\cdot\log(n\cdot k)+\log(m))}$ states and index $\mathtt{O}(n\cdot k)$, such that $\mathrm{L}(\mathcal{N}^\star) = \mathrm{L}(\langle\mathcal{A}',\mathcal{S}\rangle)$. The emptiness of $\mathcal{N}^\star$ can be checked in polynomial running-time in its number of states, exponential in its index, and linear in the alphabet size (see Theorem 5.1 of [24]). Overall, with this procedure, we obtain that the emptiness problem for an APTS is solvable in time $2^{\mathtt{O}(\log(h)+(n\cdot k)\cdot((n\cdot k)\cdot\log(n\cdot k)+\log(m)))}$. ∎

Finally, we show the computational complexity of verifying whether a given tree language, represented by a safety NPT, is recognized by an APTS.

THEOREM 5.11 (APTS-NTA Intersection Emptiness)
The *emptiness problem* for the intersection of an APTS $\langle\mathcal{A},\mathcal{S}\rangle$ with alphabet size h, where the main automaton $\mathcal{A}$ has n states and index k and the satellite $\mathcal{S}$ has m states, and a safety NTA $\mathcal{N}$ with g states, both running over B^{E}-trees, can be decided in time $g^{\mathtt{O}(n\cdot k)}\cdot 2^{\mathtt{O}(\log(h)+(n\cdot k)\cdot((n\cdot k)\cdot\log(n\cdot k)+\log(m)))}$.

PROOF. As for Theorem 5.10, the proof proceeds in two steps. First, by Theorem 5.6, there is an AATA $\mathcal{A}'$, with the same set of states and acceptance condition of $\mathcal{A}$ and a set B^{E} of directions, such that $\mathrm{L}(\mathcal{A}') = \mathrm{L}(\mathcal{A})$ and so, $\mathrm{L}(\langle\mathcal{A}',\mathcal{S}\rangle) = \mathrm{L}(\langle\mathcal{A},\mathcal{S}\rangle)$. Now, by Theorem 5.9, we obtain an NPT $\mathcal{N}^\star$, with $2^{\mathtt{O}((n\cdot k)\cdot\log(n\cdot k)+\log(m))}$ states and index $\mathtt{O}(n\cdot k)$, such that $\mathrm{L}(\mathcal{N}^\star) = \mathrm{L}(\langle\mathcal{A}',\mathcal{S}\rangle)$. Intersecting $\mathcal{N}^\star$ with $\mathcal{N}$, we obtain a new NPT $\mathcal{N}'$ such that $\mathrm{L}(\mathcal{N}') = \mathrm{L}(\langle\mathcal{A},\mathcal{S}\rangle)\cap\mathrm{L}(\mathcal{N})$, with $g\cdot 2^{\mathtt{O}((n\cdot k)\cdot\log(n\cdot k)+\log(m))}$ states and same index of $\mathcal{N}^\star$. Finally, we check the emptiness of $\mathcal{N}'$ in time $g^{\mathtt{O}(n\cdot k)}\cdot 2^{\mathtt{O}(\log(h)+(n\cdot k)\cdot((n\cdot k)\cdot\log(n\cdot k)+\log(m)))}$. ∎

6 Decision Procedures

In this section, we finally study the satisfiability and model-checking problems directly for the richer RPATL* logic, since we prove a tight 2EXPTIME upper bound for both of them.

6.1 From path formulas to satellite

As mentioned before, a RPATL* path formula is satisfied at a certain node of a path by taking into account both the future and the past. Although the past is unbounded, it only requires a finite representation. This is due to the fact that LTL with past-time operators (pLTL, for short) [17, 28] can be translated into automata on infinite words of bounded size [46]. Here, we show how to build the satellite that represents the memory on the past in order to solve satisfiability and model-checking for RPATL*, of which pLTL represents the temporal path core, as LTL is the corresponding one for ATL*.

To this aim, we first introduce the following notation, where φ is a state formula in *enf*: $\mathrm{AP}_\varphi \triangleq \mathrm{AP} \cup \mathsf{cl}(\varphi)$, $\mathrm{AP}^r_\varphi \triangleq \mathrm{AP} \cup \mathsf{rcl}(\varphi)$, and $\mathrm{AP}^{prs}_\varphi \triangleq \mathrm{AP}^r_\varphi \cup \{\mathsf{present}\}$. Intuitively, we are enriching the set of atomic propositions AP, to be used as input symbols of the automata, with the basic formulas of φ and the special proposition present.

Before showing the full satellite construction, we first describe how to build it from a single basic formula $b = \langle\!\langle A_b \rangle\!\rangle \psi_b$. Let $\widehat{\psi_b}$ be the pLTL formula obtained by replacing in ψ_b all occurrences of a direct basic subformula $b' \in \mathsf{rcl}(b)$ with the fresh label b' read as atomic proposition. It is well known that every LTL formula can be translated into a nondeterministic Büchi word automaton recognizing all infinite sequences over the sets of atomic propositions that satisfy the formula itself [48]. An extended Vardi-Wolper construction can be done for pLTL as well, as showed in [46]. Here, however, we need to build a universal automaton. Therefore, instead of applying the construction directly to the pLTL formula $\widehat{\psi_b}$, we do it for its negation $\neg\widehat{\psi_b}$ and then interpret the resulting automaton as a universal co-Büchi one. In this way, we translate $\widehat{\psi_b}$ into an automaton $\mathcal{U}_b = \langle 2^{\mathrm{AP}^{prs}_b}, \mathrm{Q}_b, \delta_b, \mathrm{Q}_{0b}, \mathrm{F}_b \rangle$, with a number of states at most exponential in $\mathsf{lng}(\psi_b)$, *i.e.*, $|\mathrm{Q}_b| = 2^{\mathsf{O}(\mathsf{lng}(\psi_b))}$, that accepts all and only the infinite words on $2^{\mathrm{AP}^{prs}_b}$ that are models of $\widehat{\psi_b}$. At this point, by applying the classical subset construction to $\mathcal{U}_b$ [41], we obtain the satellite $\mathcal{D}_b = \langle 2^{\mathrm{AP}^r_b}, 2^{\mathrm{Q}_b}, \zeta_b, \mathrm{Q}_{0b} \rangle$, where $\zeta_b(p, \sigma) \triangleq \bigcup_{q \in p} \delta_b(q, \sigma)$, for all states $p \subseteq \mathrm{Q}_b$ and labels $\sigma \subseteq \mathrm{AP}^r_b$.

To better understand the usefulness of the satellite $\mathcal{D}_b$, consider $\mathcal{U}_b$ after a prefix $\rho = \varpi_{\leq i}$ of an infinite word $\varpi \in (2^{\mathrm{AP}^r_b})^\omega$ is read. Since $\mathcal{U}_b$ is universal, there exists a number of active states that are ready to continue with the evaluation of the remaining part $\varpi_{>i}$ of the word ϖ. Consider now the satellite $\mathcal{D}_b$ after that the same prefix ρ has been read. Since $\mathcal{D}_b$ is deterministic, there is only one active state that, by construction, is exactly the set of all the active states of $\mathcal{U}_b$. It is clear then that, using $\mathcal{D}_b$, we are able to maintain together all possible parallel computations of $\mathcal{U}_b$.

We now define the product-satellite that maintains, at the same time, a memory for all path formulas ψ_b contained in a basic subformula $b \in \mathsf{cl}(\varphi)$ of the RPATL* formula φ of interest.

DEFINITION 6.1 (Memory Satellite)
The *memory satellite* for a state formula φ is the satellite $\mathcal{S}_\varphi \triangleq \langle 2^{\mathrm{AP}_\varphi}, \mathrm{P}_\varphi, \zeta_\varphi, \mathsf{p}_{0\varphi} \rangle$ isomorph to the product $\prod_{b \in \mathsf{cl}(\varphi)} \mathcal{D}_b$, where:

- the states in $\mathrm{P}_\varphi \triangleq \{\mathsf{p} \in \mathsf{cl}(\varphi) \to \bigcup_{b \in \mathsf{cl}(\varphi)} 2^{\mathrm{Q}_b} : \forall b \in \mathsf{cl}(\varphi).\, \mathsf{p}(b) \subseteq \mathrm{Q}_b\}$ are represented by the functions assigning to each basic formula b in φ a subset of the states of the associated co-Büchi word automaton $\mathcal{U}_b$, in other words, a state of $\mathcal{D}_b$;
- $\mathsf{p}_{0\varphi}(b) \triangleq \mathrm{Q}_{0b}$, for all $b \in \mathsf{cl}(\varphi)$, *i.e.*, the initial state simply maps each b to the initial states of $\mathcal{D}_b$;
- $\zeta_\varphi(\mathsf{p}, \sigma)(b) \triangleq \zeta_b(\mathsf{p}(b), \sigma \cap \mathrm{AP}^r_b)$, for all $\mathsf{p} \in \mathrm{P}_\varphi$, $\sigma \subseteq \mathrm{AP}_\varphi$, and $b \in \mathsf{cl}(\varphi)$, *i.e.*, ζ_φ collects the transition function ζ_b of $\mathcal{D}_b$, for each b.

Intuitively, this satellite records the temporal evolution of the formula φ from the root of the tree model by means of its states, which maps each basic subformula $b \in \mathsf{cl}(\varphi)$ to a set of active states of the related word automaton $\mathcal{U}_b$. Consequently, when we have to verify the RPATL* subformula $\langle\!\langle A_b \rangle\!\rangle \psi_b$ at a given point of a model, we can just do what has to be done in the case of an ATL* formula $\langle\!\langle A_b \rangle\!\rangle \psi'_b$, where ψ'_b represents the part of ψ_b still to check from that point onward, which can be identified by means of the active state of the satellite $\mathcal{D}_b$ contained into $\mathcal{S}_\varphi$. Note that the size of the latter is doubly-exponential in $\mathsf{lng}(\varphi)$, *i.e.*, its number of states is $2^{2^{\mathsf{O}(\mathsf{lng}(\varphi))}}$.

To illustrate the above construction, consider the RATL* formula $\varphi \triangleq \neg\langle\!\langle \mathrm{Ag} \rangle\!\rangle \mathsf{F}\,(\neg p \wedge \neg\langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2))$ in *enf*, which is equivalent to the one used in the example of the previous section. It is easy to see that $\mathsf{cl}(\varphi) = \{b_1, b_2\}$, where $b_1 = \langle\!\langle \mathrm{Ag} \rangle\!\rangle \mathsf{F}\,(\neg p \wedge \neg\langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2))$ and $b_2 = \langle\!\langle A \rangle\!\rangle \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2)$, from which we derive $\widehat{\psi_{b_1}} = \mathsf{F}\,(\neg p \wedge \neg b_2)$ and $\widehat{\psi_{b_2}} = \mathsf{F}\,(p_1 \wedge \mathsf{F}\, p_2)$. By applying to the latter formulas the variation of the Vardi-Wolper construction as described before, we obtain the universal co-Büchi word automata $\mathcal{U}_{b_1} = \langle 2^{\mathrm{AP}^{prs}_{b_1}}, \mathrm{Q}_{b_1}, \delta_{b_1}, \mathrm{Q}_{0b_1}, \mathrm{F}_{b_1} \rangle$ and $\mathcal{U}_{b_2} = \langle 2^{\mathrm{AP}^{prs}_{b_2}}, \mathrm{Q}_{b_2}, \delta_{b_2}, \mathrm{Q}_{0b_2}, \mathrm{F}_{b_2} \rangle$ defined as follows:

- $\mathrm{AP} = \{p, p_1, p_2\}$;
- $\mathrm{AP}^{prs}_{b_1} = \mathrm{AP} \cup \{b_2, \mathsf{present}\}$, $\mathrm{Q}_{b_1} = \{q_0, q_1\}$, and $\mathrm{Q}_{0b_1} = \mathrm{F}_{b_1} = \{q_0\}$;
- $\delta_{b_1}(q_0, \sigma) = \begin{cases} q_1, & \text{if } p \notin \sigma \text{ and } b_2 \notin \sigma; \\ q_0, & \text{otherwise;} \end{cases}$
- $\delta_{b_1}(q_1, \sigma) = q_1$;
- $\mathrm{AP}^{prs}_{b_2} = \mathrm{AP} \cup \{\mathsf{present}\}$, $\mathrm{Q}_{b_2} = \{q_0, q_1, q_2\}$, $\mathrm{Q}_{0b_2} = \{q_0\}$, and $\mathrm{F}_{b_2} = \{q_0, q_1\}$;
- $\delta_{b_2}(q_0, \sigma) = \begin{cases} q_2, & \text{if } p_1 \in \sigma \text{ and } p_2 \in \sigma; \\ q_1, & \text{if } p_1 \in \sigma \text{ and } p_2 \notin \sigma; \\ q_0, & \text{otherwise;} \end{cases}$
- $\delta_{b_2}(q_1, \sigma) = \begin{cases} q_2, & \text{if } p_2 \in \sigma; \\ q_1, & \text{otherwise;} \end{cases}$
- $\delta_{b_2}(q_2, \sigma) = q_2$.

Intuitively, by transiting to the state q_1 not contained in the co-Büchi set F_{b_1}, the automaton $\mathcal{U}_{b_1}$ checks for the existence of a letter $\sigma \in 2^{\mathrm{AP}^{prs}_{b_1}}$ in the input word containing neither p nor b_2. Similarly, $\mathcal{U}_{b_2}$ transits to the state q_2 not contained in F_{b_2}, when it finds in the input word two, not necessarily distinct, consecutive letters $\sigma_1, \sigma_2 \in 2^{\mathrm{AP}^{prs}_{b_2}}$ such that $p_1 \in \sigma_1$ and $p_2 \in \sigma_2$. Observe that, due to the very simple nature of the LTL specifications $\widehat{\psi_{b_1}}$ and $\widehat{\psi_{b_2}}$, the corresponding automata are deterministic. Consequently, the derived satellites $\mathcal{D}_{b_1}$ and $\mathcal{D}_{b_2}$ are isomorph to $\mathcal{U}_{b_1}$ and $\mathcal{U}_{b_2}$, respectively, modulo the absence of the special atomic proposition $\mathsf{present}$. Finally, we have that $\mathcal{S}_\varphi$ is isomorph to $\mathcal{D}_{b_1} \times \mathcal{D}_{b_2}$.

6.2 Satisfiability

We now describe a satisfiability procedure for RPATL*, which technically extends the one used in [42] for ATL* along with that for mCTL* proposed in [25]. Such an extension is possible since the relentful quantification has no direct interaction with the strategic feature of the logic, because it only requires that the path property ψ of a basic formula $\langle\!\langle A \rangle\!\rangle \psi$ is verified

starting from the root of the model. Indeed, as for ATL*, every CGS satisfying a RPATL* formula φ can be transformed into an *explicit* CGT model of φ itself. Such a kind of structure includes a certificate for both the truth of each basic subformula $b \in \mathsf{cl}(\varphi)$ in the respective nodes of the tree and the strategy used by the agents in A_b to achieve the goal described by the corresponding path formula ψ_b (see Section 3 of [42] for the formal definition). Observe that this result can be shown by means of the same proof of Theorem 2 of [42]. The unique difference here resides in the fact that the *witness* of a basic formula b does not start at the node from which the path formula ψ_b needs to be satisfied, *i.e.*, the root of the CGT, but from the one in which the quantification is applied, *i.e.*, the present node. Obviously, this difference directly derives from the relentful feature of RPATL*.

By exploiting the fact that a formula φ has a model iff it has an explicit model, we reduce the verification of its satisfiability to the checking of the emptiness of a SATAS $\langle \mathcal{A}_\varphi, \mathcal{S}_\varphi \rangle$ that recognizes all and only the explicit models of φ. The construction of this automaton follows the one used in Theorem 4 of [42] and changes with respect to the use of the satellite $\mathcal{S}_\varphi$, which supports the main automaton $\mathcal{A}_\varphi$ whenever it needs to start with the verification of a given path formula ψ_b, with $b \in \mathsf{cl}(\varphi)$. Indeed, by using the technique developed in [25], $\mathcal{A}_\varphi$ sends to the successors of a node x in the input tree labeled with b, all the states of the universal Co-Büchi word automaton $\mathcal{U}_b$ that are active after that it has read the word starting at the root of the tree and ending in x. Obviously, the set of active states is maintained by the component $\mathcal{D}_b$ of the satellite $\mathcal{S}_\varphi$.

Formally, $\mathcal{A}_\varphi \triangleq \mathcal{A}'_\varphi \wedge \mathcal{A}_{em} \wedge \bigwedge_{b \in \mathsf{cl}(\varphi)} \mathcal{A}_b$ is built as the linear conjunction of the following three types of automata.

- $\mathcal{A}'_\varphi$ is a one-state deterministic safety automaton that checks whether the Boolean formula $\widehat{\varphi}$ holds at the root of the input tree, where $\widehat{\varphi}$ is obtained from φ by replacing each direct basic subformula $b \in \mathsf{rcl}(\varphi)$ with the corresponding fresh atomic proposition b.
- $\mathcal{A}_{em}$ is a universal safety automaton that checks whether the input tree is a well-formed explicit model (see Lemma 5 of [42] for its definition).
- $\mathcal{A}_b$ is a universal co-Büchi automaton verifying that each node of the tree labeled by the atomic proposition b actually satisfy the formula $\langle\langle A_b \rangle\rangle \psi_b \in \mathsf{cl}(\varphi)$.

Each SATA $\mathcal{A}_b = \langle \Sigma^\star_\varphi, \mathrm{Ag}, \mathrm{Q}^\star_b, \delta^\star_b, q_b, \mathrm{F}^\star_b \rangle$ is constructed by means of the universal co-Büchi word automaton $\mathcal{U}_b = \langle 2^{\mathrm{AP}^{prs}_b}, \mathrm{Q}_b, \delta_b, \mathrm{Q}_{0b}, \mathrm{F}_b \rangle$ as follows.

- The input symbols in $\Sigma^\star_\varphi \triangleq 2^{\mathrm{AP}^\star_\varphi} \times \mathrm{P}_\varphi$ are represented by the pairs of letters in $2^{\mathrm{AP}^\star_\varphi}$ and states of the satellite $\mathcal{S}_\varphi$, where the set of extended atomic proposition $\mathrm{AP}^\star_\varphi \triangleq \mathrm{AP}_\varphi \cup \mathsf{cl}(\varphi) \times \{\mathsf{new}, \mathsf{cont}\}$ is obtained by further enriching the original one with the names of the basic subformulas of φ associated with a flag in $\{\mathsf{new}, \mathsf{cont}\}$. The latter are used to ensure that the verifications of different instances of the formula $\langle\langle A_b \rangle\rangle \psi_b$ do not overlap (this is required by the definition of explicit model [42]);
- The set of states $\mathrm{Q}^\star_b \triangleq \{q_b\} \cup \mathrm{Q}_b \times \{\mathsf{new}, \mathsf{cont}\}$ is constituted by the initial state q_b together with the pairs of a state of $\mathcal{U}_b$ and a flag in $\{\mathsf{new}, \mathsf{cont}\}$.
- The co-Büchi states in $\mathrm{F}^\star_b \triangleq \mathrm{F}_b \times \{\mathsf{cont}\}$ are simply the co-Büchi states of $\mathcal{U}_b$ paired with the flag cont.
- If $b \notin \sigma$ then $\delta^\star_b(q_b, (\sigma, \mathsf{p})) \triangleq ((\Box, \mathrm{Ag}), q_b)$, *i.e.*, when the current node is not labeled by b, the automaton simply propagates the state q_b on all its successors in order to continue the testing on the remaining part of the tree.

- If $b \in \sigma$ then $\delta_b^\star(q_b, (\sigma, \mathsf{p})) \triangleq ((\Box, \mathrm{Ag}), q_b) \wedge \bigwedge_{q \in \mathsf{p}(b)} \bigwedge_{q' \in \delta_b(q, \sigma \cap \mathrm{AP}_b^r \cup \{\mathsf{present}\})}((\Box, \mathrm{Ag}), (q', new))$, where $\mathsf{p}(b)$ is the state of the component $\mathcal{D}_b$ in the satellite $\mathcal{S}_\varphi$. Intuitively, if the current node is labeled by b, we have to verify that it satisfies the formula $\langle\langle A_b \rangle\rangle \psi_b$, assuming that the finite path going from the root to this node is the history of the play to be extended. Consequently, the automaton starts the verification of the part of the formula ψ_b that has still to be checked from the present moment onward. Observe that to identify the current node as the present moment, we force the internal copy of the automaton $\mathcal{U}_b$ to read the atomic proposition $\mathsf{present}$.
- If $(q, \alpha) \notin \sigma$ then $\delta_b^\star((q, \alpha), (\sigma, \mathsf{p})) \triangleq \mathfrak{t}$, *i.e.*, in case a state is not contained into the labeling, we do not need to verify the corresponding property on the subtree rooted on the current node.
- If $(q, \alpha) \in \sigma$ then $\delta_b^\star((q, \alpha), (\sigma, \mathsf{p})) \triangleq \bigwedge_{q' \in \delta_b(q, \sigma \cap \mathrm{AP}_b^r)}((\Box, \mathrm{Ag}), (q', \mathsf{cont}))$, *i.e.*, if a state is contained in the labeling of the current node, we have to continue the verification process by propagating on its successors all states derived from the transition function δ_b of $\mathcal{U}_b$.

One can easily note that the construction of $\mathcal{A}_b$ is almost identical to the one described in Lemma 6 of [42], but on the definition of $\delta_b^\star(q_b, (\sigma, \mathsf{p}))$ with $b \in \sigma$, which uses the satellite to determine the states of $\mathcal{U}_b$ from which we have to start the verification of the path formula ψ_b.

To better understand how the above construction works, consider again the basic formula b_2 of the previous example, together with the associated automaton $\mathcal{U}_{b_2}$ and satellite $\mathcal{D}_{b_2}$. Moreover, suppose that $\mathcal{A}_{b_2}$ is reading a node x labeled by b_2, whose corresponding track starting from the root determines a word over $\mathrm{AP}_\varphi^\star$ having at least one occurrence of the atomic proposition p_1 but none of p_2. Consequently, the satellite $\mathcal{D}_{b_2}$ identifies q_1 as the only active state of $\mathcal{U}_{b_2}$ after that the previous word has been read. So, in dependence of the labeling of x, $\mathcal{A}_{b_2}$ has to send to all of its successors one between the states (q_1, new) and (q_2, new).

Putting the above reasonings all together, we obtain the following result.

THEOREM 6.2 (RPATL* Satisfiability)
Given an RPATL* formula φ, we can build a Co-Büchi SATAS $\langle \mathcal{A}_\varphi, \mathcal{S}_\varphi \rangle$, where $\mathcal{A}_\varphi$ and $\mathcal{S}_\varphi$ have, respectively, $2^{\mathtt{O}(\mathsf{lng}(\varphi))}$ and $2^{2^{\mathtt{O}(\mathsf{lng}(\varphi))}}$ states, such that $\mathrm{L}(\langle \mathcal{A}_\varphi, \mathcal{S}_\varphi \rangle)$ is exactly the set of all the tree models of φ.

By using Theorems 6.2 and 5.10, we obtain that the check for the existence of a model for a given RPATL* specification φ can be done in time $2^{2^{\mathtt{O}(\mathsf{lng}(\varphi))}}$, resulting in a 2EXPTIME algorithm in the length of φ. Since RPATL* linearly subsumes mCTL*, which has a 2EXPTIME-HARD satisfiability problem [25], we then derive the following result.

THEOREM 6.3 (RPATL* Satisfiability Complexity)
The satisfiability problem for RPATL* is 2EXPTIME-COMPLETE.

6.3 Model checking

We finally propose a top-down model-checking algorithm for the new logic RPATL*, which checks whether the initial state of the CGS under examination satisfies the formula of interest. In particular, our procedure is inspired to the one used for mCTL* in [25] and so, it differs from that described in [1] for ATL*, which is bottom-up and uses a global model-checking approach.

With more details, by applying the standard translation of a model into an equivalent automaton [26], from a CGS $\mathcal{G}$ and a RPATL* formula φ, we easily construct a safety NTA

$\mathcal{N}_{\mathcal{G},\varphi}$ that recognizes all the extended unwindings of $\mathcal{G}$ itself, in which each state is also labeled by the basic subformulas $b \in \mathsf{cl}(\varphi)$ of φ that are true in that state. Observe that this automaton is simply linear in the size of $\mathcal{G}$. Then, by using Theorems 5.6 and 5.11, we calculate the product of $\mathcal{N}_{\mathcal{G},\varphi}$ with the SATAS of Theorem 6.2, obtaining in this way an automaton that is empty iff the model $\mathcal{G}$ does not satisfy the specification φ.

Now, by a simple calculation, we derive that the whole procedure takes time $\|\mathcal{G}\|^{2^{\mathsf{O}(\mathsf{lng}(\varphi))}}$, resulting in an algorithm that is in PTIME with respect to the size of $\mathcal{G}$ and in 2EXPTIME with respect to the size of φ. Since, by Item 1 of Theorem 4.1, there is a linear translation from ATL* to RPATL* and ATL* has a model-checking problem that is PTIME-HARD with respect to $\mathcal{G}$ and 2EXPTIME-HARD with respect to φ [1], we then derive the following result.

THEOREM 6.4 (RPATL* Model Checking Complexity)
The RPATL* model checking problem is PTIME-COMPLETE with respect to the size of the model and 2EXPTIME-COMPLETE with respect to the size of the specification.

7 Discussion, and Future Work

In this paper we have introduced RATL*, a memoryful extension of ATL*. We have studied its expressive power and its succinctness, with respect to ATL*, as well as its related decision problems. Specifically, we have shown that RATL* is equivalent but at least exponentially more succinct than ATL*. Moreover, both the satisfiability and the model-checking problems for RATL* are 2EXPTIME-COMPLETE, as they are for ATL*. Thus, this useful extension comes, in theory, at no cost. We have also investigated the extension of ATL* and RATL* with past operators (*i.e.*, backward modalities), respectively named PATL* and RPATL*. We have shown that PATL* (and thus RPATL*) is equivalent to RATL* and, as the latter, it is at least exponentially more succinct than ATL*. Then, we have shown that the complexity results we got for RATL* hold for RPATL* as well.

As for mCTL*, the interesting properties shown for RATL* make this logic not only useful at its own, but also advantageous to efficiently decide other logics (once it is shown a tight reduction to it). In the case of mCTL*, we recall that this logic is useful to decide the *embedded* CTL* *logic*, recently introduced in [38]. This logic allows to quantify over good and bad system executions. In [38], the authors also introduce a new model-checking methodology, which allows to group the system executions as good and bad, with respect to the satisfiability of a base LTL specification. By using an embedded CTL* specification, this model-checking algorithm allows checking not only whether the base specification holds or fails to hold in a system, but also how it does so. In [38], the authors use a polynomial translation of their logic into mCTL* to solve efficiently its decision problems. In the context of coalition logics, the use of an "embedded" framework seems even more interesting. In particular, an embedded ATL* logic could allow to quantify coalition of agents over good and bad system executions. Analogously to the CTL* case, one may show a polynomial translation from embedded ATL* to RATL* and use this result to efficiently solve the related decision problems.

In [5, 3, 4], *Graded Computation-Tree Logic* (GCTL, for short) has been introduced as a modal logic that extends CTL by replacing the universal (A) and existential (E) quantifiers with their graded versions $\mathsf{A}^{<n}$ and $\mathsf{E}^{\geq n}$. It has been shown that, despite such extension is strictly more expressive than CTL, the satisfiability problem for GCTL is EXPTIME-COMPLETE, as it is for CTL, even in the case that the graded numbers are coded in binary. Graded modalities have been also investigated in case of backward modalities in [6, 7]. It would be interesting

to lift the graded framework into RATL* and RPATL*, and investigate both the expressive power and the complexities of the classical decision problems for the extended logics. To give an intuition, the graded extension of RATL* can be obtained by replacing the universal ($[\![A]\!]$) and existential ($\langle\langle A\rangle\rangle$) strategy quantifiers of the logic with graded modalities of the form $[\![A]\!]^{<n}$ and $\langle\langle A\rangle\rangle^{\geq n}$. Informally speaking, these two operators have the meaning of "there exists at least n different non-equivalent strategies ..." and "for all except at most n non-equivalent strategies ..." respectively (see [] for some related material). Additionally, in the past modalities, we can predicate with a number of non-equivalent strategies in the past. Despite this extension is natural and most of the reasonings introduced in GCTL can be lifted to the new logics, there is a deep work to do regarding the formalization of equivalence among strategies.

Recently, a logic more expressive than ATL*, named Strategy Logic (SL, for short), has been introduced in [34]. The aim of this logic is to get a powerful framework for reasoning explicitly about strategic behaviors [11] in multi-agent concurrent games, by using first-order quantifications over strategies. Although SL model checking is non-elementary and the satisfiability even undecidable, there is a useful syntactic fragment of this logic, named One-Goal Strategy Logic (SL[1G], for short), which strictly subsumes ATL* and has both the above mentioned decision problems 2EXPTIME-COMPLETE, thus not harder than those for ATL* [30, 31, 32, 33]. Analogously to RATL*, one can investigate memoryful extensions of SL[1G]. Such extensions can translate to the multi-coalition framework, represented by the alternation of strategy quantifiers, the advantages of having a memoryful verification of temporal properties. This would be very important in the field of multi-agent planning and we aim to investigate this as future work.

Related works

We report that the authors of [16] have considered a sublogic of Strategy Logic, named ESL, which is orthogonal to SL[1G]. This logic uses a quantification over the history of the game, in which it is embedded a concept of memoryful quantification. Their aim was to propose a suitable framework for the synthesis of multi-player systems with rational agents. However, it is worth noting that the semantics of ESL is quite different form that one we use for RATL* and the two logics turn to be incomparable. In particular, ESL does not allow the requantification over paths as instead RATL* does (*e.g.*, ESL cannot express RATL* formulas such as $\langle\langle A\rangle\rangle \mathsf{F}\, [\![B]\!] \mathsf{G}\, p$). In addition, RATL* is able to express in its framework the ESL history quantification. For example, consider the property "for every history of the game, player 1 has a strategy that forces player 2 to satisfy ψ". Moreover, ESL requires to use a quantification over history variables, while in RATL* this property simply becomes $\mathsf{AG}\, \langle\langle 1\rangle\rangle \psi$. Finally, we observe that in [16], it is only addressed and solved the synthesis problem, while here we address and solve the satisfiability and the model-checking problems. Observe that their algorithm does not imply any result about ESL satisfiability, since they do not provide any bound on the width of ESL models. In particular, we can assert that such a bound in general does not exit, since it does not exist for SL, as it has been shown in [34] and the proof used there can be easily lifted to ESL. Consequently and similarly to SL, we can also assert that ESL satisfiability is undecidable.

In [49] a first-order variant of RPATL* has been also introduced and named FORPATL*. As in our framework, this logic allows to assert that, given any finite system-event history, no matter what future events are initiated by the an agent, the remaining agents are able

to ensure that the history can be extended to an infinite trace that satisfies a given property. Additionally, such a property is based on first order relations, with the aim to formalize a privacy policy. Clearly, FORPATL* strongly extends RPATL* and sharply refines the notion of strong compliance introduced in [2], by allowing agents to be either adversaries or cooperative. Indeed, we recall that in the classic strong compliance, the former is not allowed.

Recently, in [9, 10] a "*no-forget*" variation of the semantics of ATL and ATL* has been introduced. In a way similar to RATL*, this semantics allows agents do not forget any of their past observation in nested games. The two formalizations result to be equivalent as the past in the no-forget semantics is used in a point by agents to choose the more appropriate strategies to be used for the future evolution of the game, as we also do. This new semantics has been named in [9] *truly perfect recall* as opposed to the *perfect recall* one, classically used for ATL*. It has been investigated under the *perfect information* setting, in which the agents can observe the full state of the system (as we do), as well as under the *imperfect information* scenario [44, 23], in which only a part of a state is visible to the agents. As for RATL*, it has been shown that, under the perfect information setting, the no-forget semantics and the classical one coincide. Conversely, it has been shown that under the imperfect information setting, the two semantics are incomparable. An important consequence of the latter result is that the truly perfect recall semantics, by means of the imperfect information restriction, allows to characterize a new interesting class of games as well as to formalize interesting properties they hold. More precisely, in [10], the two semantics have been evaluated by comparing the sets of *validities* they induce, that is, the general game properties they can specify. In the validity reasoning, each formula is interpreted as a *property of interaction* between agents in a CGS. Thus, validities are properties that universally hold on the game (see [8] for more on this argument). By comparing validity sets of the classical and truly perfect recall semantics, one can compare the general properties of the class of games induced by these two semantics. Specifically, under the imperfect information, it has been shown in [10] that the truly perfect recall formally describes the specific interesting class of ATL* games in which "*players do not forget the past*". A similar reasoning can be extended to RPATL* easily and therefore this logic can be usefully used to represent and reasoning about that specific class of games. As a final observation, we report that neither in [9] nor in [10] decision problems have been investigated.

A Mathematical Notation

In this short reference appendix, we report the classical mathematical notation and some common definitions that are used along the whole work.

Classic objects We consider $\mathbb{N}$ as the set of *natural numbers* and $[m,n] \triangleq \{k \in \mathbb{N} : m \leq k \leq n\}$, $[m,n[\triangleq \{k \in \mathbb{N} : m \leq k < n\}$, $]m,n] \triangleq \{k \in \mathbb{N} : m < k \leq n\}$, and $]m,n[\triangleq \{k \in \mathbb{N} : m < k < n\}$ as its *interval* subsets, with $m \in \mathbb{N}$ and $n \in \widehat{\mathbb{N}} \triangleq \mathbb{N} \cup \{\omega\}$, where ω is the *numerable infinity*, i.e., the *least infinite ordinal*. Given a *set* X of *objects*, we denote by $|\mathrm{X}| \in \widehat{\mathbb{N}} \cup \{\infty\}$ the *cardinality* of X, i.e., the number of its elements, where ∞ represents a *more than countable* cardinality, and by $2^{\mathrm{X}} \triangleq \{\mathrm{Y} : \mathrm{Y} \subseteq \mathrm{X}\}$ the *powerset* of X, i.e., the set of all its subsets.

Relations By $R \subseteq \mathrm{X} \times \mathrm{Y}$ we denote a *relation* between the *domain* $\mathsf{dom}(R) \triangleq \mathrm{X}$ and *codomain* $\mathsf{cod}(R) \triangleq \mathrm{Y}$, whose *range* is indicated by $\mathsf{rng}(R) \triangleq \{y \in \mathrm{Y} : \exists x \in \mathrm{X}.(x,y) \in R\}$. We use $R^{-1} \triangleq \{(y,x) \in \mathrm{Y} \times \mathrm{X} : (x,y) \in R\}$ to represent the *inverse* of R itself. Moreover, by $S \circ R$, with $R \subseteq \mathrm{X} \times \mathrm{Y}$ and $S \subseteq \mathrm{Y} \times \mathrm{Z}$, we denote the *composition* of R with S, i.e.,

the relation $S \circ R \triangleq \{(x,z) \in \mathrm{X} \times \mathrm{Z} : \exists y \in \mathrm{Y}.\, (x,y) \in R \wedge (y,z) \in S\}$. We also use $R^n \triangleq R^{n-1} \circ R$, with $n \in [1,\omega[$, to indicate the *n-iteration* of $R \subseteq \mathrm{X} \times \mathrm{Y}$, where $\mathrm{Y} \subseteq \mathrm{X}$ and $R^0 \triangleq \{(y,y) : y \in \mathrm{Y}\}$ is the *identity* on Y. With $R^+ \triangleq \bigcup_{n=1}^{<\omega} R^n$ and $R^* \triangleq R^+ \cup R^0$ we denote, respectively, the *transitive* and *reflexive-transitive closure* of R. Finally, for an *equivalence* relation $R \subseteq \mathrm{X} \times \mathrm{X}$ on X, we represent with $(\mathrm{X}/R) \triangleq \{[x]_R : x \in \mathrm{X}\}$, where $[x]_R \triangleq \{x' \in \mathrm{X} : (x,x') \in R\}$, the *quotient* set of X w.r.t. R, i.e., the set of all related equivalence *classes* $[\cdot]_R$.

Functions We use the symbol $\mathrm{Y}^{\mathrm{X}} \subseteq 2^{\mathrm{X}\times\mathrm{Y}}$ to denote the set of *total functions* f from X to Y, i.e., the relations $\mathsf{f} \subseteq \mathrm{X} \times \mathrm{Y}$ such that for all $x \in \mathsf{dom}(\mathsf{f})$ there is exactly one element $y \in \mathsf{cod}(\mathsf{f})$ such that $(x,y) \in \mathsf{f}$. Often, we write $\mathsf{f} : \mathrm{X} \to \mathrm{Y}$ and $\mathsf{f} : \mathrm{X} \rightharpoonup \mathrm{Y}$ to indicate, respectively, $\mathsf{f} \in \mathrm{Y}^{\mathrm{X}}$ and $\mathsf{f} \in \bigcup_{\mathrm{X}' \subseteq \mathrm{X}} \mathrm{Y}^{\mathrm{X}'}$. Regarding the latter, note that we consider f as a *partial function* from X to Y, where $\mathsf{dom}(\mathsf{f}) \subseteq \mathrm{X}$ contains all and only the elements for which f is defined. Given a set Z, by $\mathsf{f}_{\restriction \mathrm{Z}} \triangleq \mathsf{f} \cap (\mathrm{Z} \times \mathrm{Y})$ we denote the *restriction* of f to the set $\mathrm{X} \cap \mathrm{Z}$, i.e., the function $\mathsf{f}_{\restriction \mathrm{Z}} : \mathrm{X} \cap \mathrm{Z} \rightharpoonup \mathrm{Y}$ such that, for all $x \in \mathsf{dom}(\mathsf{f}) \cap \mathrm{Z}$, it holds that $\mathsf{f}_{\restriction \mathrm{Z}}(x) = \mathsf{f}(x)$. Moreover, with $\varnothing$ we indicate a generic *empty function*, i.e., a function with empty domain. Note that $\mathrm{X} \cap \mathrm{Z} = \emptyset$ implies $\mathsf{f}_{\restriction \mathrm{Z}} = \varnothing$. Finally, for two partial functions $\mathsf{f}, \mathsf{g} : \mathrm{X} \rightharpoonup \mathrm{Y}$, we use $\mathsf{f} \Cup \mathsf{g}$ and $\mathsf{f} \Cap \mathsf{g}$ to represent, respectively, the *union* and *intersection* of these functions defined as follows: $\mathsf{dom}(\mathsf{f} \Cup \mathsf{g}) \triangleq \mathsf{dom}(\mathsf{f}) \cup \mathsf{dom}(\mathsf{g}) \setminus \{x \in \mathsf{dom}(\mathsf{f}) \cap \mathsf{dom}(\mathsf{g}) : \mathsf{f}(x) \neq \mathsf{g}(x)\}$, $\mathsf{dom}(\mathsf{f} \Cap \mathsf{g}) \triangleq \{x \in \mathsf{dom}(\mathsf{f}) \cap \mathsf{dom}(\mathsf{g}) : \mathsf{f}(x) = \mathsf{g}(x)\}$, $(\mathsf{f} \Cup \mathsf{g})(x) = \mathsf{f}(x)$ for $x \in \mathsf{dom}(\mathsf{f} \Cup \mathsf{g}) \cap \mathsf{dom}(\mathsf{f})$, $(\mathsf{f} \Cup \mathsf{g})(x) = \mathsf{g}(x)$ for $x \in \mathsf{dom}(\mathsf{f} \Cup \mathsf{g}) \cap \mathsf{dom}(\mathsf{g})$, and $(\mathsf{f} \Cap \mathsf{g})(x) = \mathsf{f}(x)$ for $x \in \mathsf{dom}(\mathsf{f} \Cap \mathsf{g})$.

Words By X^n, with $n \in \mathbb{N}$, we denote the set of all *n-tuples* of elements from X, by $\mathrm{X}^* \triangleq \bigcup_{n=0}^{<\omega} \mathrm{X}^n$ the set of *finite words* on the *alphabet* X, by $\mathrm{X}^+ \triangleq \mathrm{X}^* \setminus \{\varepsilon\}$ the set of *non-empty words*, and by X^ω the set of *infinite words*, where, as usual, $\varepsilon \in \mathrm{X}^*$ is the *empty word*. The *length* of a word $w \in \mathrm{X}^\infty \triangleq \mathrm{X}^* \cup \mathrm{X}^\omega$ is represented with $|w| \in \widehat{\mathbb{N}}$. By $(w)_i$ we indicate the *i-th letter* of the finite word $w \in \mathrm{X}^+$, with $i \in [0, |w|[$. Furthermore, by $\mathsf{fst}(w) \triangleq (w)_0$ (resp., $\mathsf{lst}(w) \triangleq (w)_{|w|-1}$), we denote the *first* (resp., *last*) letter of w. In addition, by $(w)_{\leq i}$ (resp., $(w)_{>i}$), we indicate the *prefix* up to (resp., *suffix* after) the letter of index i of w, i.e., the finite word built by the first $i+1$ (resp., last $|w|-i-1$) letters $(w)_0, \ldots, (w)_i$ (resp., $(w)_{i+1}, \ldots, (w)_{|w|-1}$). We also set, $(w)_{<0} \triangleq \varepsilon$, $(w)_{<i} \triangleq (w)_{\leq i-1}$, $(w)_{\geq 0} \triangleq w$, and $(w)_{\geq i} \triangleq (w)_{>i-1}$, for $i \in [1, |w|[$. Mutatis mutandis, the notations of i-th letter, first, prefix, and suffix apply to infinite words too. Finally, by $\mathsf{pfx}(w_1, w_2) \in \mathrm{X}^\infty$ we denote the *maximal common prefix* of two different words $w_1, w_2 \in \mathrm{X}^\infty$, i.e., the finite word $w \in \mathrm{X}^*$ for which there are two words $w_1', w_2' \in \mathrm{X}^\infty$ such that $w_1 = w \cdot w_1'$, $w_2 = w \cdot w_2'$, and $\mathsf{fst}(w_1') \neq \mathsf{fst}(w_2')$. By convention, we set $\mathsf{pfx}(w,w) \triangleq w$.

Trees For a set Δ of objects, called *directions*, a *Δ-tree* is a set $\mathrm{T} \subseteq \Delta^*$ closed under prefix, i.e., if $t \cdot d \in \mathrm{T}$, with $d \in \Delta$, then also $t \in \mathrm{T}$. We say that it is *complete* if it holds that $t \cdot d' \in \mathrm{T}$ whenever $t \cdot d \in \mathrm{T}$, for all $d' < d$, where $< \subseteq \Delta \times \Delta$ is an a priori fixed strict total order on the set of directions that is clear from the context. Moreover, it is *full* if $\mathrm{T} = \Delta^*$. The elements of T are called *nodes* and the empty word ε is the *root* of T. For every $t \in \mathrm{T}$ and $d \in \Delta$, the node $t \cdot d \in \mathrm{T}$ is a *successor* of t in T. The tree is *b-bounded* if the maximal number b of its successor nodes is finite, i.e., $b = \max_{t \in \mathrm{T}} |\{t \cdot d \in \mathrm{T} : d \in \Delta\}| < \omega$. A *branch* of the tree is an infinite word $w \in \Delta^\omega$ such that $(w)_{\leq i} \in \mathrm{T}$, for all $i \in \mathbb{N}$. For a finite set Σ of objects, called *symbols*, a *Σ-labeled Δ-tree* is a quadruple $\langle \Sigma, \Delta, \mathrm{T}, \mathsf{v} \rangle$, where T is a Δ-tree and $\mathsf{v} : \mathrm{T} \to \Sigma$ is a *labeling function*. When Δ and Σ are clear from the context, we

call $\langle \mathrm{T}, \mathsf{v} \rangle$ simply a (labeled) tree.

B Full Definition of RPATL* Syntax and Semantics

The syntax of RPATL* is formally defined as follows.

DEFINITION B.1
RPATL* *state* (φ) and *path* (ψ) *formulas* are built inductively from the sets of atomic propositions AP and agents Ag using the following context-free grammar, where $p \in \mathrm{AP}$ and $A \subseteq \mathrm{Ag}$:

1. $\varphi ::= \mathsf{present} \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle\!\langle A \rangle\!\rangle \psi \mid [\![A]\!] \psi$;
2. $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathsf{X}\,\psi \mid \mathsf{Y}\,\psi \mid \tilde{\mathsf{Y}}\,\psi \mid \psi\,\mathsf{U}\,\psi \mid \psi\,\mathsf{S}\,\psi \mid \psi\,\mathsf{R}\,\psi \mid \psi\,\mathsf{B}\,\psi$.

The class of RPATL* formulas is the set of all the state formulas generated by the above grammar, in which the occurrences of the special atomic proposition *present* is in the scope of a strategy quantifier.

The semantics of RPATL* is formally defined as follows.

DEFINITION B.2
Given a CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, two initial traces $\rho, \rho_p \in \mathrm{Trk}(s_0)$, a path $\pi \in \mathrm{Pth}(s_0)$, and a number $k \in \mathbb{N}$, it holds that:

1. $\mathcal{G}, \rho, \rho_p \models \mathsf{present}$ iff $\rho = \rho_p$;
2. $\mathcal{G}, \rho, \rho_p \models p$, for $p \in \mathrm{AP}$, iff $p \in \lambda(\mathsf{lst}(\rho))$;
3. $\mathcal{G}, \rho, \rho_p \models \neg\varphi$ iff not $\mathcal{G}, \rho, \rho_p \models \varphi$, that is $\mathcal{G}, \rho, \rho_p \not\models \varphi$;
4. $\mathcal{G}, \rho, \rho_p \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{G}, \rho, \rho_p \models \varphi_1$ and $\mathcal{G}, \rho, \rho_p \models \varphi_2$;
5. $\mathcal{G}, \rho, \rho_p \models \varphi_1 \vee \varphi_2$ iff $\mathcal{G}, \rho, \rho_p \models \varphi_1$ or $\mathcal{G}, \rho, \rho_p \models \varphi_2$;
6. $\mathcal{G}, \rho, \rho_p \models \langle\!\langle A \rangle\!\rangle \psi$ if there exists a $\mathsf{lst}(\rho)$-total strategy $\mathsf{f}_A \in \mathrm{Str}(A, \mathsf{lst}(\rho))$ such that, for all plays $\pi \in \mathrm{Play}(\mathsf{f}_A)$, it holds that $\mathcal{G}, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$;
7. $\mathcal{G}, \rho, \rho_p \models [\![A]\!] \psi$ if, for all $\mathsf{lst}(\rho)$-total strategies $\mathsf{f}_A \in \mathrm{Str}(A, \mathsf{lst}(\rho))$, there exists a play $\pi \in \mathrm{Play}(\mathsf{f}_A)$ such that $\mathcal{G}, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$;

Moreover, for a path π, and a number $k \in \mathbb{N}$, it holds that:

8. $\mathcal{G}, \pi, k, \rho_p \models \varphi$ iff $\mathcal{G}, \pi_{\leq k}, \rho_p \models \varphi$;
9. $\mathcal{G}, \pi, k, \rho_p \models \neg\psi$ iff not $\mathcal{G}, \pi, k, \rho_p \models \psi$, that is $\mathcal{G}, \pi, k, \rho_p \not\models \psi$;
10. $\mathcal{G}, \pi, k, \rho_p \models \psi_1 \wedge \psi_2$ iff $\mathcal{G}, \pi, k, \rho_p \models \psi_1$ and $\mathcal{G}, \pi, k, \rho_p \models \psi_2$;
11. $\mathcal{G}, \pi, k, \rho_p \models \psi_1 \vee \psi_2$ iff $\mathcal{G}, \pi, k, \rho_p \models \psi_1$ or $\mathcal{G}, \pi, k, \rho_p \models \psi_2$;
12. $\mathcal{G}, \pi, k, \rho_p \models \mathsf{X}\,\psi$ iff $\mathcal{G}, \pi, k+1, \rho_p \models \psi$;
13. $\mathcal{G}, \pi, k, \rho_p \models \mathsf{Y}\,\psi$ iff $k > 0$ and $\mathcal{G}, \pi, k-1, \rho_p \models \psi$;
14. $\mathcal{G}, \pi, k, \rho_p \models \tilde{\mathsf{Y}}\,\psi$ iff $k = 0$ or $\mathcal{G}, \pi, k-1, \rho_p \models \psi$;
15. $\mathcal{G}, \pi, k, \rho_p \models \psi_1 \mathsf{U}\,\psi_2$ iff there is an index i, with $k \leq i$, such that $\mathcal{G}, \pi, i, \rho_p \models \psi_2$ and, for all indexes j, with $k \leq j < i$, it holds $\mathcal{G}, \pi, j, \rho_p \models \psi_1$;
16. $\mathcal{G}, \pi, k, \rho_p \models \psi_1 \mathsf{S}\,\psi_2$ iff there is an index i, with $i \leq k$, such that $\mathcal{G}, \pi, i, \rho_p \models \psi_2$ and, for all indexes j, with $i < j \leq k$, it holds $\mathcal{G}, \pi, j, \rho_p \models \psi_1$;

17. $\mathcal{G}, \pi, k, \rho_p \models \psi_1 \mathsf{R}\, \psi_2$ iff for all indexes i, with $k \leq i$, it holds that $\mathcal{G}, \pi, i, \rho_p \models \psi_2$ or there is an index j, with $k \leq j < i$, such that $\mathcal{G}, \pi, j, \rho_p \models \psi_1$;
18. $\mathcal{G}, \pi, k, \rho_p \models \psi_1 \mathsf{B}\, \psi_2$ iff for all indexes i, with $i \leq k$, it holds that $\mathcal{G}, \pi, i, \rho_p \models \psi_2$ or there is an index j, with $i < j \leq k$, such that $\mathcal{G}, \pi, j, \rho_p \models \psi_1$;

References

[1] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[2] A. Barth, A. Datta, J.C. Mitchell, and H. Nissenbaum. Privacy and Contextual Integrity: Framework and Applications. In *IEEE Symposium on Security and Privacy'06*, pages 184–198. IEEE Computer Society, 2006.

[3] A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic. In *IEEE Symposium on Logic in Computer Science'09*, pages 342–351. IEEE Computer Society, 2009.

[4] A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic with Binary Coding. In *EACSL Annual Conference on Computer Science Logic'10*, LNCS 6247, pages 125–139. Springer, 2010.

[5] A. Bianco, F. Mogavero, and A. Murano. Graded Computation Tree Logic. *ACM Transactions On Computational Logic*, 13(3):25:1–53, 2012.

[6] P.A. Bonatti, C. Lutz, A. Murano, and M.Y. Vardi. The Complexity of Enriched μ-caluli. In *ICALP (2)*, LNCS 4052, pages 540–551. Springer, 2006.

[7] P.A. Bonatti, C. Lutz, A. Murano, and M.Y. Vardi. The Complexity of Enriched Mu-Calculi. *Logical Methods in Computer Science*, 4(3):1–27, 2008.

[8] N. Bulling and W. Jamroga. Comparing Variants of Strategic Ability: How Uncertainty and Memory Influence General Properties of Games. *Autonomous Agents and Multi-Agent Systems*, 28(3):474–518, 2014.

[9] N. Bulling, W. Jamroga, and M. Popovici. Agents With Truly Perfect Recall in Alternating-Time Temporal Logic. In *Autonomous Agents and Multiagent Systems'14*, pages 1561–1662, 2014.

[10] N. Bulling, W. Jamroga, and M. Popovici. ATL* With Truly Perfect Recall: Expressiveness and Validities. In *European Conference on Artificial Intelligence'14*, page 6, 2014.

[11] K. Chatterjee, T.A. Henzinger, and N. Piterman. Strategy Logic. *Information and Computation*, 208(6):677–693, 2010.

[12] M. Daniele, P. Traverso, and M.Y. Vardi. Strong Cyclic Planning Revisited. In *European Conference on Planning'99*, pages 35–48, 2000.

[13] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time. *Journal of the ACM*, 33(1):151–178, 1986.

[14] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[15] M.J. Fischer and R.E. Ladner. Propositional Dynamic Logic of Regular Programs. *Journal of Computer and System Science*, 18(2):194–211, 1979.

[16] D. Fisman, O. Kupferman, and Y. Lustig. Rational Synthesis. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems'10*, LNCS 6015, pages 190–204. Springer, 2010.

[17] D.M. Gabbay. The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In *Temporal Logic in Specification'87*, LNCS 398, pages 409–448. Springer, 1987.

[18] X. Huang. Bounded Planning for Strategic Goals with Incomplete Information and Perfect Recall. In *Autonomous Agents and Multiagent Systems'13*, pages 885–892, 2013.

[19] W. Jamroga. Strategic Planning Through Model Checking of ATL Formulae. In *International Conference on Artificial Intelligence and Soft Computing'04*, LNCS 3070, pages 879–884. Springer, 2004.

[20] D. Janin and I. Walukiewicz. Automata for the Modal μ-Calculus and Related Results. In *International Symposiums on Mathematical Foundations of Computer Science'95*, LNCS 969, pages 552–562. Springer, 1995.

[21] O. Kupferman and A. Pnueli. Once and For All. In *IEEE Symposium on Logic in Computer Science'95*, pages 25–35. IEEE Computer Society, 1995.

[22] O. Kupferman, A. Pnueli, and M.Y. Vardi. Once and For All. *Journal of Computer and System Science*, 78(3):981–996, 2012.

[23] O. Kupferman and M.Y. Vardi. Module Checking Revisited. In *CAV*, LNCS 1254, pages 36–47. Springer, 1997.

[24] O. Kupferman and M.Y. Vardi. Weak Alternating Automata and Tree Automata Emptiness. In *ACM Symposium on Theory of Computing'98*, pages 224–233, 1998.

[25] O. Kupferman and M.Y. Vardi. Memoryful Branching-Time Logic. In *IEEE Symposium on Logic in Computer Science'06*, pages 265–274. IEEE Computer Society, 2006.

[26] O. Kupferman, M.Y. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time Model Checking. *Journal of the ACM*, 47(2):312–360, 2000.

[27] F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal Logic with Forgettable Past. In *IEEE Symposium on Logic in Computer Science'02*, pages 383–392. IEEE Computer Society, 2002.

[28] O. Lichtenstein, A. Pnueli, and L.D. Zuck. The Glory of the Past. In *Logic of Programs'85*, pages 196–218, 1985.

[29] F. Mogavero. Branching-Time Temporal Logics (Theoretical Issues and a Computer Science Application). Master's thesis, Universitá degli Studi di Napoli "Federico II", Napoli, Italy, October 2007.

[30] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning About Strategies: On the Model-Checking Problem. Technical Report 1112.6275, arXiv, December 2011.

[31] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. A Decidable Fragment of Strategy Logic. Technical Report 1202.1309, arXiv, February 2012.

[32] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic. In *CONCUR*, LNCS 7454, pages 193–208. Springer, 2012.

[33] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning About Strategies: On the Model-Checking Problem. *ACM Transactions On Computational Logic*, 15(4):34:1–42, 2014.

[34] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning About Strategies. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science'10*, LIPIcs 8, pages 133–144, 2010.

[35] F. Mogavero, A. Murano, and M.Y. Vardi. Relentful Strategic Reasoning in Alternating-Time Temporal Logic. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning'10*, LNAI 6355, pages 371–387. Springer, 2010.

[36] D.E. Muller and P.E. Schupp. Alternating Automata on Infinite Trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.

[37] D.E. Muller and P.E. Schupp. Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of Theorems of Rabin, McNaughton, and Safra. *Theoretical Computer Science*, 141(1-2):69–107, 1995.

[38] P. Niebert, D. Peled, and A. Pnueli. Discriminative Model Checking. In *Computer Aided Verification'08*, LNCS 5123, pages 504–516. Springer, 2008.

[39] M.J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[40] M. Pistore and M.Y. Vardi. The Planning Spectrum - One, Two, Three, Infinity. *Journal of Artificial Intelligence Research*, 30:101–132, 2007.

[41] M.O. Rabin and D.S. Scott. Finite Automata and their Decision Problems. *IBM Journal of Research and Development*, 3:115–125, 1959.

[42] S. Schewe. ATL* Satisfiability is 2ExpTime-Complete. In *International Colloquium on Automata, Languages and Programming'08*, LNCS 5126, pages 373–385. Springer, 2008.

[43] S. Schewe and B. Finkbeiner. Satisfiability and Finite Model Property for the Alternating-Time μ-Calculus. In *EACSL Annual Conference on Computer Science Logic'06*, LNCS 4207, pages 591–605. Springer, 2006.

[44] P.Y. Schobbens. Alternating-Time Logic with Imperfect Recall. *ENTCS*, 85(2):82–93, 2004.

[45] W. van der Hoek and M.J. Wooldridge. Tractable Multiagent Planning for Epistemic Goals. In *Autonomous Agents and Multiagent Systems'02*, pages 1167–1174, 2002.

[46] M.Y. Vardi. A Temporal Fixpoint Calculus. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages'88*, pages 250–259, 1988.

[47] M.Y. Vardi. Reasoning about The Past with Two-Way Automata. In *ICALP*, LNCS 1443, pages 628–641. Springer, 1998.

[48] M.Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *IEEE Symposium on Logic in Computer Science'86*, pages 332–344. IEEE Computer Society, 1986.

[49] W.H. Winsborough, J. von Ronne, O. Chowdhury, J. Niu, and Md.S. Ashik. Towards Practical Privacy Policy Enforcement. Technical Report CS-TR-2011-009, The University of Texas at San Antonio, 2011.

[50] M.J. Wooldridge. *Introduction to Multiagent Systems.* John Wiley & Sons, 2001.

Received ...