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

Fabio Mogavero, *Università degli Studi di Napoli "Federico II", I-80126 Napoli, Italy.*

Aniello Murano, *Università degli Studi di Napoli "Federico II", I-80126 Napoli, Italy.*

Moshe Y. Vardi, *Rice University, Department of Computer Science, Houston, TX 77251-1892, 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 their history.

In this paper, we define mATL*, a memoryful extension of ATL*, in which a formula is satisfied at a certain node of a path by taking into account both the future and the past. We study the expressive power of mATL*, its succinctness, as well as related decision problems. We also investigate the relationship between memoryful quantification and past modalities and show their equivalence. We show that both the memoryful 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 they are for ATL*.

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

November 27, 2012

1 Introduction

Multi-agent concurrent systems recently emerged as a new paradigm for better understanding distributed systems [11, 41]. In this kind of systems, different processes can have different goals and the interactions between them may be adversarial or cooperative. Thus the latter can be seen as games in the classical framework of game theory, with adversarial coalitions [32]. Classical branching-time temporal logics, such as CTL* [10], 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

[1] This work is partially based on the paper [27], which appeared in LPAR'10.

concurrent 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 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 [11], 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 agents in A such that, no matter the strategy that is executed by agents not in A, the resulting outcome of the interaction in $\mathcal{G}$ satisfies ψ at s. Coming back to the previous example, one can see that the property p can be expressed by the ATL* formula $\langle\langle\{1,2\}\rangle\rangle\mathsf{G}\,\neg fail$, 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 [19]. 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* [9], 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 [33]. 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 [19] 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 obtained from CTL* by simply adding a special proposition *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 [15, 37]. In such a framework, a memoryful enhancement of ATL* enables "relentful" planning, that is, agents can relent and change their goals, depending on their history[1]. That is, 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 mean that agents change their strategy, but that they can choose a strategy that allows them to change their goals. For example, consider the ATL* formula $\langle\langle\emptyset\rangle\rangle\mathsf{G}\,\langle\langle A\rangle\rangle\psi$. In the memoryful framework, this formula is satisfied by a CGS $\mathcal{G}$ (at its starting node) iff for each possible trace (history) ρ the agents in A can ensure that the evolution of $\mathcal{G}$ that extends ρ satisfies ψ from the start state.

In this paper, we introduce and study the logic mATL*, a memoryful extension of ATL*. Thus, mATL* can be thought of as a fusion of mCTL* and ATL* in a common framework. Similarly to mCTL*, the syntax of mATL* is obtained from ATL* by simply adding a special proposition *present*. Semantically, mATL* is obtained from ATL* by reinterpreting the path quantifiers of the logic to be memoryful. More specifically, for a CGS $\mathcal{G}$, the mATL* 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

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

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. As an example of the usefulness of the relentful reasoning, consider 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 $\langle\langle A\rangle\rangle(\mathsf{F}\,(q \vee r) \wedge \mathsf{G}\,f)$, 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 change their goal to satisfy u until v holds from the *start* of the computation. This cannot be easily handled in ATL*, since the specification depends on the past. On the other hand, it can be handled in mATL*, with the formula $\langle\langle A\rangle\rangle((p\,\mathsf{U}\,(q \vee r)) \wedge \mathsf{G}\,f)$, where f is $r \rightarrow \langle\langle A\rangle\rangle(u\,\mathsf{U}\,v)$.

In the paper, we also consider an extension of mATL* with *past operators* (mpATL*, for short). As for classical temporal and modal logics, past operators allow reasoning about the past in a computation [6, 7, 22, 23, 38]. In mpATL*, we can further require that coalitions of agents had a memoryful 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 example, by using P as the dual of F, we can change the alternative goal f of agents in A to be $r \rightarrow \mathsf{P}\,(h \wedge \langle\langle A\rangle\rangle(u\,\mathsf{U}\,v))$, which requires that once r occurs at a state s, at a previous state s' of s in which h holds, the subformula u until v from the start of the computation must be true.

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 [19] for mCTL*. Since memoryfulness refers to behavior from the start of the computation, which occurred in the past, memoryfulness is intimately connected to the past. Indeed, we prove this formally. We study the expressive power and the succinctness of mATL* w.r.t ATL*, as well as the memoryless fragment of mpATL* (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 mATL* and pATL* are at least exponentially more succinct than ATL*. As for m$^-$ATL* (where the minus stands for the variant of the logic without the "present" proposition, but the path interpretation is still memoryful), we prove that it is strictly less expressive than ATL*. On the other hand, we prove that pATL* is equivalent to p$^-$ATL*, but exponentially more succinct.

From an algorithmic point of view, we examine, for mpATL*, the two classical decision problems: *model checking* and *satisfiability*. We show that model checking is not easier than satisfiability and in particular that both 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* [20]. 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, the complex *symmetric alternating tree automata with satellites* (SATAS, for short), which extends both *automata over concurrent game structures* in [36] and *alternating automata with satellites* in [19], in a common setting. For technical convenience, the states of the whole automaton are partitioned into states regarding the satellite and those regarding the rest of the automaton, which we call the *main automaton*. The complexity results then come from the fact that mpATL* 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 w.r.t. 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 plays, strategies, plays, and unwinding. Then, we have Section 3, in which we introduce mATL* and define its syntax and semantics, followed by Section 4, in which it is defined the extension mpATL* and there are studied the expressiveness and succinctness relationship of both the logics. 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 mATL* and mpATL*. Note that, in the accompanying Appendix A, we recall standard mathematical notation and some basic definitions that are used in the paper.

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.

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 $\mathcal{G}$ w.r.t. 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 the (resp., s-total) strategies of agents in A.

A path π in $\mathcal{G}$ starting at a state s is a *play* w.r.t. 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 Memoryful 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 memoryful quantification over paths, in a similar way it has been done for the memoryful branching-time temporal logic mCTL* [19].

3.1 Syntax

The *memoryful alternating-time temporal logic* (mATL*, 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 for agents in A"*, respectively. The syntax of mATL* 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 mATL* formulas can refer to a special proposition $\mathsf{present}$, which enables us to refer to the present time. Readers familiar with mCTL* can see mATL* as mCTL* where strategy quantifiers substitute path quantifiers. The formal syntax of mATL* follows.

DEFINITION 3.1 (mATL* Syntax)
mATL* *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. $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi \mid \psi\,\mathsf{R}\,\psi$.

mATL* is the set of all state formulas generated by the above grammar, in which the occurrences of the special proposition $\mathsf{present}$ 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*. 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* [12] of ψ defined recursively as for ATL* in the following way: $\mathsf{cl}(\varphi) \triangleq \{\varphi\} \cup \mathsf{cl}'(\varphi)$, for all *basic formulas* $\varphi = \mathsf{Qn}\ \psi$, with $\mathsf{Qn} \in \{\langle\langle A\rangle\rangle, [\![A]\!]\}$, and $\mathsf{cl}(\psi) \triangleq \mathsf{cl}'(\psi)$, in all other cases, where *(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)* $\mathsf{cl}'(\mathsf{Qn}\ \psi) \triangleq \mathsf{cl}(\psi)$, 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 mATL* is defined w.r.t. concurrent game structures. However, the two logics differ on interpreting state formulas. First, in mATL* 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 mATL* 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 *memoryful 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 mATL* 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 agent in A. Thus, when evaluating path formulas in mATL* 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 w.r.t. states and paths in the structure, respectively. In mATL*, instead, we add an additional parameter, the *present track*, which is the track that led from the initial state to the point of quantification. Path formulas are again evaluated w.r.t. paths, but state formulas are now evaluated w.r.t. tracks, which are viewed as partial executions.

We now formally define mATL* semantics w.r.t. 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 being the present. The semantics of mATL* state formulas involving $\neg$, $\wedge$, and $\vee$, as well as that for mATL* path formulas, except for the state formula case, is defined as usual in ATL*. The semantics of the remaining part, which involves the memoryful feature, follows.

DEFINITION 3.2 (mATL* Semantics)
Given a CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, two initial traces $\rho, \rho_p \in \mathrm{Trc}(s_0)$, 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 $[\![A]\!] \psi$ "reset the present", i.e., their satisfaction w.r.t 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 mATL* 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 $\rho, \rho_p \in \mathrm{Trc}(\mathcal{G}, s_0)$, 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 $\neg\langle\langle x \rangle\rangle \varphi \equiv [\![x]\!] \neg\varphi$.

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 mATL*.

THEOREM 3.3 (mATL* Unwinding Invariance)
mATL* 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. As first thing, 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 partial order on tracks.

At this point, we prove the statement by showing 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 $\pi \in \mathrm{Pth}_{\mathcal{G}_U}(\varepsilon)$, $k \in \mathbb{N}$, and $\rho_p \in \mathrm{Trk}_{\mathcal{G}_U}(\varepsilon)$, such that $\rho_p < \pi$.

We now prove, by induction on the structure of formulas, the three cases of special proposition 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 definition of semantics, 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 definition of unw_{trk}, we have that $\mathsf{lst}(\mathsf{unw}_{trk}(\rho)) = \mathsf{unw}(\mathsf{lst}(\rho))$. Thus, by definition of the unwinding function unw, it holds that $\lambda_{\mathcal{G}}(\mathsf{lst}(\mathsf{unw}_{trk}(\rho))) = \lambda_{\mathcal{G}_U}(\mathsf{lst}(\rho))$. At this point, 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 $\mathcal{G}, \mathsf{unw}_{trk}(\rho), \mathsf{unw}_{trk}(\rho_p) \models p$. Therefore, $\mathcal{G}_U, \rho, \rho_p \models p$ 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 definition of semantics, 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 $\mathcal{G}, \mathsf{unw}_{pth}(\rho \cdot \pi_{\geq 1}), 0, \mathsf{unw}_{trk}(\rho) \models \psi$. Now,to prove the statement, we have only 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 $\mathsf{h} : \mathrm{Trk}_{\mathcal{G}}(s') \rightharpoonup \overline{\mathrm{Trk}}_{\mathcal{G}_U}(s)$ 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 $\mathsf{d} \triangleq (\mathsf{f}_A(\mathsf{h}(\rho')), \mathsf{d}^c_A)$.

  At this point, we can define the strategy $\mathsf{f}'_A \in \mathrm{Str}_{\mathcal{G}}(A, s')$ as follows: $\mathsf{f}'_A(\rho') \triangleq \mathsf{f}_A(\mathsf{h}(\rho'))$, for all $\rho' \in \mathsf{dom}(\mathsf{f}'_A) \triangleq \mathsf{dom}(\mathsf{h})$. Now, 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 definition of semantics, 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 $\mathcal{G}, \mathsf{unw}_{trk}(\rho) \cdot \pi'_{\geq 1}, 0, \mathsf{unw}_{trk}(\rho) \models \psi$. Now, define the strategy $\mathsf{f}_A \in \mathrm{Str}_{\mathcal{G}_U}(A, s)$ as follows: $\mathsf{f}_A(\rho) \triangleq \mathsf{f}'_A(\mathsf{unw}_{trk}(\rho))$, for all $\rho \in \mathrm{Trk}_{\mathcal{G}_U}(s)$. At this point, it is easy to see that, for all plays $\pi \in \mathrm{Play}_{\mathcal{G}_U}(\mathsf{f}_A)$, it holds that $\mathsf{unw}_{pth}(\pi) \in \mathrm{Play}_{\mathcal{G}}(\mathsf{f}'_A)$. Therefore, $\mathcal{G}, \mathsf{unw}_{trk}(\rho) \cdot \mathsf{unw}_{pth}(\pi)_{\geq 1}, 0, \mathsf{unw}_{trk}(\rho) \models \psi$, i.e., $\mathcal{G}, \mathsf{unw}_{pth}(\rho \cdot \pi_{\geq 1}), 0, \mathsf{unw}_{trk}(\rho) \models \psi$. Now, by the inductive hypothesis, it holds that $\mathcal{G}_U, \rho \cdot \pi_{\geq 1}, 0, \rho \models \psi$. Hence, 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 mATL* also enjoys the tree model property.

COROLLARY 3.4 (mATL* Tree Model Property)
mATL* 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 mATL* 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 w.r.t. $\mathtt{L}_1$.

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, 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 mATL* w.r.t. ATL* as well as some extensions/restrictions of mATL*. In particular, we consider the logics mpATL* and pATL* to be, respectively, mATL* and ATL* augmented with the past-time operators *"previous"* and *"since"*, which dualize the future-time operators *"next"* and *"until"* as in pLTL [22] and pCTL* [17]. 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 m$^-$ATL*, p$^-$ATL*, and mp$^-$ATL* to be, respectively, the syntactical restriction of mATL*, pATL*, and mpATL* in which the use of the atomic proposition present is not allowed. On one hand, we have that all mentioned logics are expressively equivalent, except for m$^-$ATL* and p$^-$ATL*. 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 φ', but φ' 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 [14]. All the discussed results are reported in the following theorem.

THEOREM 4.1 (Reductions)
The following properties hold:

1. ATL* (resp., pATL*) is linearly reducible to mATL* (resp., mpATL*);
2. mpATL* (resp., mp$^-$ATL*) is linearly reducible to pATL* (resp., p$^-$ATL*);
3. mpATL* (resp., mp$^-$ATL*) is non-elementarily reducible to mATL* (resp., m$^-$ATL*);
4. pATL* is non-elementarily reducible to ATL*;
5. m$^-$ATL* and p$^-$ATL* are at least exponentially more succinct than ATL*;
6. m$^-$ATL* is less expressive then ATL*.

PROOF. Let φ be an input formula for items 1-4.

- Items 1 and 2 follow by replacing each subformula $\langle\langle A\rangle\rangle\psi$ in φ by $\langle\langle A\rangle\rangle \mathsf{F}\,(\mathsf{present} \wedge \psi)$ and $\langle\langle A\rangle\rangle \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge \psi)$, respectively, where $\mathsf{P}\,\psi'$ is the corresponding past-time operator for $\mathsf{F}\,\psi'$ and $\tilde{\mathsf{Y}}\,\psi'$ is the weak previous time operator, which is true if either ψ' is true in the previous time-step or such a time-step does not exist. Note that all the formula substitutions start from the innermost subformula.
- Item 3 follows by replacing each subformula $\langle\langle A\rangle\rangle\psi$ in φ by $\langle\langle A\rangle\rangle\psi'$, where ψ' is obtained by the Separation Theorem (see Theorem 2.4 of [14]), which allows to eliminate all pure-past formulas[2]. Note that, as in the above items, the substitutions start from the innermost subformula. Moreover, the non-elementary blow-up is inherited from the use of the Separation Theorem.
- Item 4 proceeds as for the translation of pCTL* into CTL* (see Lemma 3.3 and Theorem 3.4 of [17]). The only difference here is that, when we apply the Separation Theorem to obtain a path formula as a disjunction of formulas of the form $ps \wedge pr \wedge ft$, where ps, pr, and ft are respectively pure-past, pure-present (i.e., Boolean combinations of atomic propositions and basic formulas), and pure-future formulas, we need to substitute the $\mathsf{present}$ proposition with $\mathfrak{f}$ in ps and ft and with $\mathfrak{t}$ in pr. As for the previous item, the origin of the non-elementary blow-up resides in the Separation Theorem.
- Item 5 follows by using the formula $\varphi \triangleq \langle\langle A\rangle\rangle \mathsf{G}\,(\bigwedge_{i=1}^{n}(p_i \leftrightarrow [\![\emptyset]\!]p_i) \rightarrow (p_0 \leftrightarrow [\![\emptyset]\!]p_0))$ (resp., $\varphi \triangleq \langle\langle A\rangle\rangle \mathsf{G}\,(\bigwedge_{i=1}^{n}(p_i \leftrightarrow \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge p_i)) \rightarrow (p_0 \leftrightarrow \mathsf{P}\,((\tilde{\mathsf{Y}}\,\mathfrak{f}) \wedge p_0))))$, which is similar to that used to prove that pLTL is exponentially more succinct than LTL (see Theorem 3.1 of [21]). By using an argument similar to that used in [21], we obtain the desired result.
- Item 6 follows by using a proof similar to that used for m$^-$CTL* (see Theorem 3.4 of [19]), and so showing that the ATL formula $\varphi \triangleq \langle\langle A\rangle\rangle \mathsf{F}\,((\![\emptyset]\!]\mathsf{X}\,p) \wedge (\![\emptyset]\!]\mathsf{X}\,\neg p))$ has no m$^-$ATL* equivalent formula.

∎

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

COROLLARY 4.2 (Expressiveness)
mATL*, p$^-$ATL*, pATL*, and mpATL* have the same expressive power of ATL*. m$^-$ATL* and mp$^-$ATL* have the same expressive power, but are less expressive than ATL*. Moreover, all of them are at least exponentially more succinct than ATL*.

[2] A pure-past formula contains only past-time operators. In Item 4, we also consider pure-future formulas, which contain only future-time operators, and pure-present formulas, which do not contain any temporal operator at all.

Figure 1 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.

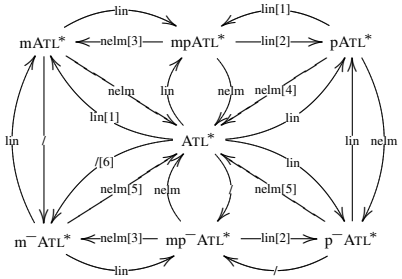

FIG. 1: Expressive power and succinctness hierarchy.

5 Alternating Tree Automata

In this section, we briefly introduce an automaton model used to solve efficiently the satisfiability and model-checking problems for mpATL*, 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 mpATL*, as it is stated in Theorem 3.3.

5.1 Classic automata

Alternating tree automata [29] 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* [16] 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 [36].

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* (SATA, 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* (AATA, 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 an 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 $\mathrm{S} \subseteq \Delta \times \mathrm{Q}$ 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 $(q', x \cdot (d, d')) \in \mathsf{l}(y)$;
- 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 $(q', x \cdot (d, d')) \in \mathsf{l}(y)$;

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 $\mathsf{l}(y) \triangleq \{\mathsf{r}(y \cdot j) : j \in \mathbb{N} \wedge y \cdot j \in \mathrm{R}\}$ 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 [20], 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.

Let $\mathcal{R} = \langle \mathrm{R}, \mathsf{r} \rangle$ be a run of an ATA $\mathcal{A}$ on a tree $\mathcal{T}$ and $\mathrm{R}' \subseteq \mathrm{R}$ one of its branches. Then, by $\mathsf{inf}(\mathrm{R}') \triangleq \{q \in \mathrm{Q} : |\{y \in \mathrm{R}' : \mathsf{r}(y) = q\}| = \omega\}$ 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.

At this point, we can 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 is 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 $\mathrm{L}(\mathcal{A}) = \emptyset$ or not.

Now, we show how to reduce, for equivalence, a SATA to an AATA when it is known a priori 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 that 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. At this point, it is immediate to see that the thesis follows directly by Definition 5.3 of SATA run. ∎

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 [19]. 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 satisfiability 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 problem.

We now formally define this new fundamental concept 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 an SATA (resp., AATA) and $\mathcal{S} \triangleq \langle \Sigma, \mathrm{P}, \zeta, p_0 \rangle$ is a *deterministic safety word automaton*, a.k.a. *satellite*, where P is a non-empty finite set of *states*, $p_0 \in \mathrm{P}$ is an *initial states*, 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 $\langle \mathcal{A}, \mathcal{S} \rangle$.

At this point, we can define the language 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 consider, in particular, 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 that has 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 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 $\mathrm{L}(\mathcal{N}^\star) = \mathrm{L}(\langle \mathcal{A}, \mathcal{S} \rangle)$.

PROOF. To deduce the thesis, we use the Muller-Schupp exponential-time nondeterminization procedure [30] 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 $\mathrm{L}(\langle \mathcal{N}, \mathcal{S} \rangle) = \mathrm{L}(\langle \mathcal{A}, \mathcal{S} \rangle)$. At this point, 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 [19], 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 [36]), 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. Hence, 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 definition of ATAS, we obtain that $\mathrm{L}(\langle \mathcal{A}', \mathcal{S} \rangle) = \emptyset$ iff $\mathrm{L}(\langle \mathcal{A}, \mathcal{S} \rangle) = \emptyset$. At this point, 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 $\mathrm{L}(\mathcal{N}^\star) = \mathrm{L}(\langle \mathcal{A}', \mathcal{S} \rangle)$. Now, 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 [18]). 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 how much costs to verify if 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 n' states, both running over B^{E}-trees, can be decided in time $n'^{\mathtt{O}(n\cdot k)} \cdot 2^{\mathtt{O}(\log(h)+(n\cdot k)\cdot((n\cdot k)\cdot\log(n\cdot k)+\log(m)))}$.

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 $n' \cdot 2^{\mathtt{O}((n\cdot k)\cdot\log(n\cdot k)+\log(m))}$ states and same index of $\mathcal{N}^\star$. Finally, we check the emptiness of $\mathcal{N}'$ in time $n'^{\mathtt{O}(n\cdot k)} \cdot 2^{\mathtt{O}(\log(h)+(n\cdot k)\cdot((n\cdot k)\cdot\log(n\cdot k)+\log(m)))}$. ∎

6 Decision Procedures

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

6.1 From path formulas to satellite

As mentioned before, an mATL* 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 unlimited, it only requires a finite representation. This is due to the fact that LTL with past operators (pLTL, for short) [14, 22] can be translated into automata on infinite words of bounded size [39], and that it represents the temporal path core of mpATL* (as LTL is the corresponding one for ATL*). Here, we show how to build the satellite that represents the memory on the past in order to solve satisfiability and model-checking for mpATL*.

To this aim, we first introduce the following notation, where φ is an *enf* state formula: $\mathrm{AP}_\varphi = \mathrm{AP} \cup \mathsf{cl}(\varphi)$, $\mathrm{AP}^r_\varphi = \mathrm{AP} \cup \mathsf{rcl}(\varphi)$, and $\mathrm{AP}^{prs}_\varphi = \mathrm{AP}^r_\varphi \cup \{\mathsf{present}\}$. 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 the occurrences of a direct basic subformula $b' \in \mathsf{rcl}(b)$ by the label b' read as atomic proposition. By using a slight variation of the procedure developed in [39], we can translate $\widehat{\psi_b}$ into a universal co-Büchi word automaton $\mathcal{U}_b = \langle \mathrm{AP}^{prs}_b, \mathrm{Q}_b, \delta_b, \mathrm{Q}_{0b}, \mathrm{F}_b\rangle$, with a number of states at most exponential in $\mathsf{lng}(\psi_b)$, i.e., $|\mathrm{Q}_b| = 2^{\mathtt{O}(\mathsf{lng}(\psi_b))}$, that accepts all and only the infinite traces on AP^{prs}_b that are models of $\widehat{\psi_b}$. By applying the classical subset construction to $\mathcal{U}_b$ [34], we obtain the satellite $\mathcal{D}_b = \langle \mathrm{AP}^r_b, 2^{\mathrm{Q}_b}, \zeta_b, \mathrm{Q}_{0b}\rangle$, 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 that a prefix $\rho = \varpi_{\leq i}$ of an infinite trace $\varpi \in (\mathrm{AP}^r_b)^\omega$ 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 trace ϖ. Consider now the satellite $\mathcal{D}_b$ after that the same prefix ρ is 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 all possible 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 mpATL* formula φ we want to check.

DEFINITION 6.1 (Memory Satellite)

The *memory satellite* for a state formula φ is the satellite $\mathcal{S}_\varphi \triangleq \langle \mathrm{AP}_\varphi, \mathrm{P}_\varphi, \zeta_\varphi, \mathsf{p}_{0\varphi} \rangle$, where *(i)* $\mathrm{P}_\varphi \triangleq \{\mathsf{p} \in (\bigcup_{b \in \mathsf{cl}(\varphi)} 2^{\mathrm{Q}_b})^{\mathsf{cl}(\varphi)} : \forall b \in \mathsf{cl}(\varphi).\ \mathsf{p}(b) \subseteq \mathrm{Q}_b\}$, *(ii)* $\mathsf{p}_{0\varphi}(b) \triangleq \mathrm{Q}_{0b}$, and *(iii)* $\zeta_\varphi(\mathsf{p}, \sigma)(b) \triangleq \bigcup_{q \in \mathsf{p}(b)} \delta_b(q, \sigma \cap \mathrm{AP}_b^r)$, for all $\mathsf{p} \in \mathrm{P}_\varphi$, $\sigma \subseteq \mathrm{AP}_\varphi$, and $b \in \mathsf{cl}(\varphi)$.

Intuitively, this satellite record the temporal evolution of the formula φ from the root of the tree model by means of its states, which are represented by functions mapping each basic subformula $b \in \mathsf{cl}(\varphi)$ to a set of active states of the related word automaton $\mathcal{U}_b$. Note that the size of the satellite $\mathcal{S}_\varphi$ is doubly-exponential in $\mathsf{lng}(\varphi)$, i.e., its number of states is $2^{2^{\mathfrak{O}(\mathsf{lng}(\varphi))}}$.

6.2 Satisfiability

The satisfiability procedure we now propose technically extends that used for ATL* in [35] along with that for mCTL* in [19]. Such an extension is possible due to the fact that the memoryful quantification has no direct interaction with the strategic features of the logic. In particular as for ATL*, it is possible to show that every CGS model of an mpATL* formula φ can be transformed into an *explicit* CGT model of φ. Such a model includes a certificate for both the truth of each of its basic subformula $b \in \mathsf{cl}(\varphi)$ in the respective node of the tree and the strategy used by the agents A_b to achieve the goal described by the corresponding path formula ψ_b (for a formal definition see [35]). The main difference of our definition of explicit models w.r.t. that given in [35] is in the fact that the *witness* of a basic formula b does not start in the node from which the path formula ψ_b needs to be satisfied, but from the node in which the quantification is applied, i.e., the present node. This difference, which directly derives from the memoryful feature of mpATL*, is due to the request that ψ_b needs to be satisfied on a path that starts at the root of the model. The proof of an explicit model existence is exploited by constructing an SATAS that accepts all and only the explicit models of the specification. The proof follows that used in Theorem 4 of [35] and changes w.r.t. the use of the satellite $\mathcal{S}_\varphi$ that helps the main automaton $\mathcal{A}_\varphi$ whenever it needs to start with the verification of a given path formula ψ_b, with $b \in \mathsf{cl}(\varphi)$. In particular, $\mathcal{A}_\varphi$ needs to send to the successors of a node x labeled with b in the tree given in input, all the states of the universal Co-Büchi automaton $\mathcal{U}_b$ that are active after $\mathcal{U}_b$ has read the word derived by the trace starting in the root of the tree and ending in x. By extending an idea given in [19], this requirement is satisfied by $\mathcal{A}_\varphi$ by defining the transition function, for the part of interest, as follows: $\delta(q_b, (\sigma, \mathsf{p})) = ((\Box, \mathrm{Ag}), q_b) \wedge \bigwedge_{q \in \mathsf{p}(b)} \bigwedge_{q' \in \delta_b(q, \sigma \cap \mathrm{AP}_b^r \cup \{\mathsf{present}\})} ((\Box, \mathrm{Ag}), (q', new))$, where $b \in \sigma$ and $\mathsf{p}(b)$ is the component relative to b of the product-state of $\mathcal{S}_\varphi$.

Putting the above reasoning all together, the following result holds.

THEOREM 6.2 (mpATL* Satisfiability)

Given an mpATL* 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, $2^{\mathfrak{O}(\mathsf{lng}(\varphi))}$ and $2^{2^{\mathfrak{O}(\mathsf{lng}(\varphi))}}$ 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 of the existence of a model for a given mpATL* specification φ can be done in time $2^{2^{\mathfrak{O}(\mathsf{lng}(\varphi))}}$, resulting in a 2EXPTIME algorithm in the length of φ. Since mpATL* subsumes mCTL*, which has a satisfiability problem 2EXPTIME-HARD [19], we then derive the following result.

THEOREM 6.3 (mpATL* Satisfiability Complexity)

The satisfiability problem for mpATL* is 2EXPTIME-COMPLETE.

6.3 Model checking

As for mCTL*, for the new logic mpATL* we use a top-down model-checking algorithm that checks whether the initial state of the CGS under exam satisfies the formula. In particular, the procedure we propose is similar to that used for mCTL* in [19] and so, it is different from that used for ATL* in [1], which is bottom-up and uses a global model-checking method.

With more details, from the CGS $\mathcal{G}$ and an mpATL* formula φ, we easily construct a safety NTA $\mathcal{N}_{\mathcal{G},\varphi}$ that recognize all the extended unwindings of $\mathcal{G}$ itself, in which each state is also labeled by the basic subformulas $\varphi' \in \mathsf{cl}(\varphi)$ of φ that are true in that state [20]. This automaton is simply linear in the size of $\mathcal{G}$. Then, by calculating the product of $\mathcal{N}_{\mathcal{G},\varphi}$ with the SATAS of Theorem 6.2, we obtain an automata that is empty iff the model does not satisfy the specification.

Now, by a simple calculation based on the result of Theorem 5.11, we derive that the whole procedure takes time $\|\mathcal{G}\|^{2^{O(\mathsf{lng}(\varphi))}}$, resulting in an algorithm that is in PTIME w.r.t. the size of $\mathcal{G}$ and in 2EXPTIME w.r.t. the size of φ. Since, by Item 1 of Theorem 4.1, there is a linear translation from ATL* to mpATL* and ATL* has a model-checking problem that is PTIME-HARD w.r.t. $\mathcal{G}$ and 2EXPTIME-HARD w.r.t φ [1], we then derive the following result.

THEOREM 6.4 (mpATL* Model Checking Complexity)
The mpATL* model checking problem is PTIME-COMPLETE w.r.t. the size of the model and 2EXPTIME-COMPLETE w.r.t. the size of the specification.

7 Discussion and Future Work

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

As for mCTL*, the interesting properties shown for mATL* 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 [31]. This logic allows to quantify over good and bad system executions. In [31], the authors also introduce a new model-checking methodology, which allows to group the system executions as good and bad, w.r.t 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 [31], 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 mATL* and use this result to efficiently solve the related decision problems.

In [3, 4, 5], *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 [7, 6]. It would be interesting to lift the graded framework into mATL* and mpATL*, 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 mATL* 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. 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 [26]. The aim of this logic is to get a powerful framework for reasoning explicitly about strategic behaviors [8] 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* [24, 25, 28]. Analogously to mATL*, 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 [13] 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 mATL* and the two logics turn to be incomparable. In particular, ESL does not allow the requantification over paths as instead mATL* does (e.g., ESL cannot express mATL* formulas such as $\langle\!\langle A\rangle\!\rangle \mathsf{F}\, [\![B]\!] \mathsf{G}\, p$). In addition, mATL* 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 force player 2 to satisfy ψ". Moreover, ESL requires to use a quantification over history variables, while in mATL* this property simply becomes $\mathsf{AG}\, \langle\!\langle 1\rangle\!\rangle \psi$. Finally, we enlighten that in [13], 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 [26] 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.

Recently, in [40] a first-order variant of mpATL* has been also introduced and named FOmpATL*. 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, FOmpATL* strongly extends mpATL* 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.

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.

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. Association for Computing Machinery. Under publication.

[4] 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.

[5] 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.

[6] 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.

[7] Piero A. Bonatti, Carsten Lutz, Aniello Murano, and Moshe Y. Vardi. The complexity of enriched μ-caluli. In *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 540–551. Springer, 2006.

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

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

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. What makes atl* decidable? a decidable fragment of strategy logic. In *CONCUR*, volume 7454 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2012.

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

[30] 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.

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

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

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

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

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

[36] 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.

[37] 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.

[38] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.

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

[40] 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.

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

Received ...