Relentful Strategic Reasoning in Alternating-Time Temporal Logic

Fabio Mogavero[*,†]
Universitá degli Studi di Napoli "Federico II", I-80126 Napoli, Italy.
mogavero@na.infn.it
Aniello Murano[†]
Universitá degli Studi di Napoli "Federico II", I-80126 Napoli, Italy.
murano@na.infn.it
Moshe Y. Vardi[‡]
Rice University, Department of Computer Science, Houston, TX 77251-1892, U.S.A.
vardi@cs.rice.edu

Abstract

Temporal logics are a well investigated formalism for the specification, verification, and synthesis of reactive systems. Within this family, alternating temporal logic, ATL*, 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 past 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*.

1 Introduction

Multi-agent systems recently emerged as a new paradigm for better understanding distributed systems [FHMV95, Woo01]. In multi-agent systems, different processes can have different goals and the interactions between them may be adversarial or cooperative. Interactions between processes in multi-agent systems can thus be seen as games in the classical framework of game theory, with adversarial coalitions [OR94]. Classical branching-time temporal logics, such as CTL* [EH86], turn out to be of limited power when applied to multi-agent systems. For example, consider the property `Prop`: "processes 1 and 2 cooperate to ensure that a system (having more than two processes) never enters a fail state". It is well known that CTL* cannot express `Prop` [AHK02]. Rather, CTL* can only say whether the set of all agents can or cannot prevent the system from entering a fail state.

In order to allow the temporal-logic framework to work within the setting of multi-agent systems, Alur, Henzinger, and Kupferman introduced *Alternating-Time Temporal Logic* (ATL*, for short) [AHK02].

[*]Part of this research was done while the author was visiting the Rice University. Partially supported by ESF "Games for Design and Verification" project short visit grant n. 3339.

[†]Partially supported by MIUR PRIN Project n.2007-9E5KM8 and Vigevani Research Project Prize 2010.

[‡]Work supported in part by NSF grants CCF-0613889, ANI-0216467, and CCF-0728882, by BSF grant 9800096, and by gift from Intel.

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*, which can be used to represent the power that a coalition of agents has to achieve certain results. In particular, these modalities express selective quantifications over those paths that can be effected as outcomes of infinite games between the coalition and its complement. ATL* formulas are interpreted over *game structures* (closely related to *systems* in [FHMV95]), which model a set of interacting processes. Given a game structure $\mathcal{G}$ and a set A of agents, the ATL* formula $\langle\langle A \rangle\rangle \psi$ is satisfied at a state s iff there is 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 satisfies ψ at s. Coming back to the previous example, one can see that the property `Prop` can be expressed by the ATL* formula $\langle\langle \{1,2\} \rangle\rangle \mathsf{G} \neg \mathit{fail}$, where G is the classical temporal modality "*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 [KV06]. 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* [DTV00], 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 [PV07]. 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 [KV06] 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 very popular in the context of multi-agent system planning [vdHW02, Jam04]. 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 game structure $\mathcal{G}$ (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 game structure $\mathcal{G}$, 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 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 for 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*

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

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 logics, past operators allow reasoning about the past in a computation [LPZ85]. 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.

An important contribuition of this work is to show for the first time a clear and complete picture of the relationships among ATL* and its various extensions with memoriful quantification and past modalities, which goes beyond the expressiveness results obtained in [KV06] 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 two decision problems for mpATL*, *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 upper bounds, we follow an *automata-theoretic approach* [KVW00]. 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 *agent-action tree automata with satellites* (AGCTAS, for short), which extends both *automata over concurrent game structures* in [SF06] and *alternating automata with satellites* in [KV06], in a common setting. For technical convenience, AGCTAS states 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 an AGCTAS 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 AGCTAS is solvable in EXPTIME w.r.t. both the size of the main automaton and the logarithm of the size of the satellite.

As for mCTL*, the interesting properties shown for mATL* make this logic not only useful to 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 has been useful to decide the *embedded* CTL* *logic* (EmCTL*, for short), recently introduced in [NPP08]. EmCTL* allows to quantify over good and bad system executions. In [NPP08], 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 EmCTL* 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 [NPP08], the authors use a polynomial translation of EmCTL* into mCTL* to solve efficiently decision problems related to EmCTL*. In the context of coalition logics, the use of an "embedded" framework seems even more interesting. In particular, an embedded ATL* logic (EmATL*, for short) could allow to quantify coalition of agents over good and bad system executions. Analogously to EmCTL*, one may show a polynomial translation from EmATL* to mATL* and use this result to efficiently solve decision problems concerning EmATL*. We postpone the details to the full version of this paper.

The outline of the paper follows. In Section 2, we recall the basic notions regarding concurrent game structures, strategies, plays, trees, and unwinding. In Section 3, we first introduce mATL* and define its syntax and semantics. Then, we introduce its extension mpATL* and study the expressiveness and succinctness of both mATL* and mpATL*. Finally, in Section 4, we introduce AGCTAS and show how to solve the satisfiability and model-checking problems for both mATL* and mpATL*.

2 Preliminaries

A *concurrent game structure* (CGS, for short) is a tuple $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, 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*, $\lambda : \mathrm{St} \mapsto 2^{\mathrm{AP}}$ is a *labeling* function that maps each state s to the set of atomic propositions true in that state, $\tau : \mathrm{St} \times \mathrm{Ac}^{\mathrm{Ag}} \mapsto \mathrm{St}$ is a *transition* function that maps a state and a *global decision* d (i.e., a function from Ag to Ac) to a state, and $s_0 \in \mathrm{St}$ is a designated *initial state*. By $|\mathcal{G}| = |\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Ag}|}$ we denote the *size* of the $\mathcal{G}$. If the set of actions is finite, i.e., $b = |\mathrm{Ac}| < \infty$, 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*. It is easy to note that $\mathcal{G}$ is finite iff it has a finite size. For a set of agents A, a *decision* for A is $\mathsf{d}_A \in \mathrm{Ac}^A$ and a *counterdecision* for A is a decision $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag} \setminus A}$ for agents not in A. By $\mathsf{d} = (\mathsf{d}_A, \mathsf{d}^c_A)$, we denote the *composition* of d_A and d^c_A.

A *trace* (resp., a *path*) is a finite (resp., an infinite) sequence of states $\rho \in \mathrm{St}^*$ (resp., $\pi \in \mathrm{St}^\omega$) such that, for all $0 \leq i < |\rho| - 1$ (resp., $i \in \mathbb{N}$), there exists a global decision d_i such that $\rho_{i+1} = \tau(\rho_i, \mathsf{d}_i)$ (resp., $\pi_{i+1} = \tau(\pi_i, \mathsf{d}_i)$). Intuitively, traces and paths are legal sequences of reachable states. A trace ρ is said *non-empty* iff $|\rho| > 0$ and *initial* iff $\rho_0 = s_0$, i.e., if ρ starts in the initial state. Moreover, with $\pi_{\leq i}$ we indicate the *prefix* up to the state of index i of the path π, i.e., the trace built by the first $i+1$ states $\pi_0, \cdots, \pi_i$. Finally, we use $\mathrm{Trc} \subseteq \mathrm{St}^*$ to indicate the sets of all the non-empty traces.

A *strategy* for a set of agents $A \subseteq \mathrm{Ag}$ is a partial function $\mathsf{f}_A : \mathrm{Trc} \rightharpoonup \mathrm{Ac}^A$ that maps a non-empty trace ρ to a decision $\mathsf{f}_A(\rho)$ of agents in A. A strategy f_A is called *memoryless* iff all its values depend only on the last state of the trace; otherwise, it is called *memoryful*. Formally, f_A is memoryless iff, for all traces ρ and states s with $\rho \cdot s$ belonging to the domain $\mathsf{dom}(\mathsf{f}_A)$ of f_A, it holds that $\mathsf{f}_A(\rho \cdot s) = \mathsf{f}_A(s)$. For a state s, we also say that f_A is *s-defined* iff it is defined on all the non-empty traces starting in s that are reachable through f_A. Formally, f_A is s-defined if $s \in \mathsf{dom}(\mathsf{f}_A)$ and for all traces $\rho \in \mathsf{dom}(\mathsf{f}_A)$, it holds that $\rho_0 = s$ and, for all counterdecisions d^c_A, it holds that $\rho \cdot \tau(\rho_{|\rho|-1}, (\mathsf{f}_A(\rho), \mathsf{d}^c_A)) \in \mathsf{dom}(\mathsf{f}_A)$. A path π is a *play* w.r.t. a π_0-defined strategy f_A of agents in A (f_A-*play*, for short), iff for all $i \in \mathbb{N}$, there is a counterdecision $\mathsf{d}^c_{A,i}$ such that $\pi_{i+1} = \tau(\pi_i, \mathsf{d}_i)$, where $\mathsf{d}_i = (\mathsf{f}_A(\pi_{\leq i}), \mathsf{d}^c_{A,i})$.

For a set Δ, a Δ-*tree* is a prefix-closed set $\mathrm{T} \subseteq \Delta^*$, i.e., if $x \cdot x' \in \mathrm{T}$, with $x' \in \Delta$, then also $x \in \mathrm{T}$. Elements of T are *nodes* and ε is its *root*. For every $x \in \mathrm{T}$ and $x' \in \Delta$, the node $x \cdot x' \in \mathrm{T}$ is a *successor* of x in T. T is *b-bounded* if the maximal number b of its node successors is finite. For a finite set Σ, a Σ-*labeled* Δ-*tree* is a pair $\langle \mathrm{T}, \mathsf{v} \rangle$, where T is a Δ-tree and $\mathsf{v} : \mathrm{T} \mapsto \Sigma$ is a *labeling* function. We drop Δ and Σ when they are clear from the context. For a node $x = y_0 \cdots y_k \in \mathrm{T}$, we denote by $\mathsf{trcto}(x)$ and $\mathsf{wrdto}(x)$, respectively, the trace $(\varepsilon) \cdot (y_0) \cdots (y_0 \cdots y_k) \in \mathrm{T}^*$, and the word $\mathsf{v}(\varepsilon) \cdot \mathsf{v}(y_0) \cdots \mathsf{v}(y_0 \cdots y_k) \in \Sigma^*$. Finally, a Σ-*labeled agent-action tree* (AAT, for short) is a tuple $\mathcal{T} = \langle \mathrm{Ag}, \mathrm{Ac}, \mathrm{T}, \mathsf{v} \rangle$, where Ag and Ac are as in CGSs and $\langle \mathrm{T}, \mathsf{v} \rangle$ is a Σ-labeled $\mathrm{Ac}^{\mathrm{Ag}}$-tree.

A CGS $\mathcal{U} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, where St is an $\mathrm{Ac}^{\mathrm{Ag}}$-tree, $s_0 = \varepsilon$, and $\tau(s, \mathsf{d}) = s \cdot \mathsf{d}$, is called *concurrent game tree* (CGT, for short). With each CGT $\mathcal{U}$ we can associate a 2^{AP}-labeled AAT $\mathcal{T} = \langle \mathrm{Ag},

$\mathrm{Ac}, \mathrm{T}, \mathsf{v}\rangle$, in which $\mathrm{T} = \mathrm{St}$ and $\mathsf{v}(x) = \lambda(x)$, for all nodes x. Note that a b-bounded CGT has as set of states a $b^{|\mathrm{Ag}|}$-bounded tree. Given a CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$, the *unwinding* $\mathcal{U}_{\mathcal{G}}$ of $\mathcal{G}$ is the CGT $\langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}', \lambda', \tau', \varepsilon \rangle$ for which there is a surjective function $\mathsf{unw} : \mathrm{St}' \mapsto \mathrm{St}$ such that $\mathsf{unw}(\varepsilon) = s_0$ and, for all nodes x and decisions d, we have $\mathsf{unw}(x \cdot \mathsf{d}) = \tau(\mathsf{unw}(x), \mathsf{d})$ and $\lambda'(x) = \lambda(\mathsf{unw}(x))$. Note that each CGS $\mathcal{G}$ has a unique associated unwinding $\mathcal{U}_{\mathcal{G}}$ and so a unique AAT $\mathcal{T}_{\mathcal{G}}$. Finally, all above definitions of trace, path, strategy, and play easily extend to AAT.

3 Memoryful Alternating-Time Temporal Logic

In this section, we introduce the *memoryful alternating-time temporal logic* (mATL*, for short), obtained by allowing the alternating-time temporal logic ATL* [AHK02] to use memoryful quantification over paths, in a similar way it has been done for the memoryful branching-time temporal logic mCTL* [KV06]. mATL* inherits from ATL* the existential $\langle\langle A \rangle\rangle$ and the universal $[[A]]$ *strategy(-play) 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 *state formulas* 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 atomic proposition *present*, which enables us to refer to the present. Readers familiar with mCTL* can see mATL* as mCTL* where strategy quantifiers substitute path quantifiers. The formal syntax of mATL* follows.

Definition 3.1. *Let* AP *and* Ag *be the sets of atomic propositions and agents.* mATL* state *(φ) and* path *(ψ)* formulas *are built inductively by the following context-free grammar, with $p \in \mathrm{AP}$ and $A \subseteq \mathrm{Ag}$:*

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

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

The *length* $|\varphi|$ of a formula φ is defined inductively on the structure of φ itself, in the classical way, by also considering $|\langle\langle A \rangle\rangle\varphi|$ and $|[[A]]\varphi|$ to be equal to $1 + |A| + |\varphi|$.

As for ATL*, the semantics of mATL* is defined w.r.t. a concurrent game structure. However, the two logics differ on interpreting state formulas. First, in mATL* the satisfaction of a state formula is related to a specific trace, while in ATL* it is related only to a state. Moreover, path quantification in mATL* ranges over paths that start at the initial state and contain as prefix the trace that lead to the present state; we refer to this trace as the *present trace*. This is what we refer to as *memoryful quantification*. In contrast, in ATL* path quantification ranges 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 trace 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 event that preceded the point of quantification. In ATL*, state formulas are evaluated w.r.t. states in the structure and path formulas are evaluated w.r.t. paths in the structure. In mATL* we add an additional parameter, the *present trace*, which is the trace 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. traces, which are viewed as partial executions. We now formally define mATL* semantics w.r.t. a CGS $\mathcal{G}$.

For two non-empty initial traces ρ and ρ_p, where ρ_p is the present trace, 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 π, a non-empty present trace ρ_p 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 the 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 CTL* (see Appendix A, for a full definition). The semantics of the remaining part, which involves the memoryful feature, follows:

Definition 3.2. *Given a* CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$*, two initial traces* $\rho, \rho_p \in \mathrm{Trc}$*, a path* π*, and a number* $k \in \mathbb{N}$*, where* $\rho = \rho' \cdot s$*,* $\rho' \in \mathrm{Trc} \cup \{\varepsilon\}$*, and* $s \in \mathrm{St}$*, it holds that:*

1. $\mathcal{G}, \rho, \rho_p \models \mathrm{present}$ *iff* $\rho = \rho_p$*;*
2. $\mathcal{G}, \rho, \rho_p \models p$*, for* $p \in \mathrm{AP}$*, iff* $p \in \lambda(s)$*;*
3. $\mathcal{G}, \rho, \rho_p \models \langle\langle A \rangle\rangle \psi$ *iff there exists an s-defined strategy* f_A *of agents in A such that for all* f_A*-plays* π *it holds that* $\mathcal{G}, \rho' \cdot \pi, 0, \rho \models \psi$*;*
4. $\mathcal{G}, \rho, \rho_p \models [[A]] \psi$ *iff for all the s-defined strategies* f_A *of agents in A there exists an* f_A*-play* π *such that* $\mathcal{G}, \rho' \cdot \pi, 0, \rho \models \psi$*;*
5. $\mathcal{G}, \pi, k, \rho_p \models \varphi$ *iff* $\mathcal{G}, \pi_{\leq k}, \rho_p \models \varphi$.

Note that the present trace ρ_p comes into 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.

We say that a CGS $\mathcal{G}$ is a *model* of an mATL* formula φ, denoting this by $\mathcal{G} \models \varphi$, iff $\mathcal{G}, s_0, s_0 \models \varphi$. Moreover, φ is said *satisfiable* iff there exists a model $\mathcal{G}$ for it. For two mATL* formulas φ_1 and φ_2 we say that φ_1 is *equivalent* to φ_2, formally $\varphi_1 \equiv \varphi_2$, iff, for all CGSs $\mathcal{G}$, and non-empty traces ρ and ρ_p, it holds that $\mathcal{G}, \rho, \rho_p \models \varphi_1$ iff $\mathcal{G}, \rho, \rho_p \models \varphi_2$.

By induction on the syntactical structure of the sentences, it is possible to prove the following classical result. Note that this is a basic step towards the automata-theoretic approach we use to solve the model-checking and the satisfiability problems for mATL*.

Theorem 3.3. mATL* *satisfies the tree model property. In fact, for each* CGS $\mathcal{G}$ *and formula* φ*, it holds that* $\mathcal{G} \models \varphi$ *iff* $\mathcal{U}_{\mathcal{G}} \models \varphi$.

From this result and the one-to-one connection between the CGT $\mathcal{U}_{\mathcal{G}}$ (obtained as the unwinding of the CGS $\mathcal{G}$) and the related AAT $\mathcal{T}_{\mathcal{G}}$, we say that $\mathcal{T}_{\mathcal{G}}$ satisfies φ iff $\mathcal{G} \models \varphi$.

When we compare two logics, the basic comparison is in terms of *expressiveness*. A logic L_1 is *as expressive* as a logic L_2 iff every formula in L_2 is logically equivalent to some formula in L_1. If L_1 is as expressive as L_2, but there is a formula in L_1 that is not logically equivalent to any formula in L_2, then L_1 is *more expressive* than L_2. If L_1 is as expressive as L_2 and vice versa, then L_1 and L_2 are *expressively equivalent*. We can compare the logics L_1 and L_2 also in terms of *succinctness*, which measures the necessary blow-up when translating between the logics. Note that comparing logics in terms of succinctness makes sense, when the logics are not expressively equivalent, focusing then on their common fragment. In fact, a logic L_1 can be more expressive than a logic L_2, but at the same time, less succinct than the latter.

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 [LPZ85] and pCTL* [KP95] (see Appendix A, for more). 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 logic 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 nonelementary space in $|\varphi|$ (shortly, we say that pATL* is nonelementary 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 [Gab87]. All the discussed results are reported in the following theorem.

Theorem 3.4. *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 nonelementarily reducible to* mATL* *(resp.,* m¯ATL**);*
4. pATL* *is nonelementarily 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**.*

Sketch. 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 $\langle\langle A\rangle\rangle \mathsf{F}\,(present \wedge \psi)$ and $\langle\langle A\rangle\rangle \mathsf{P}\,((\widetilde{\mathsf{Y}}\,\mathfrak{false}) \wedge \psi)$, respectively, where $\mathsf{P}\,\psi'$ is the corresponding past-time operator for $\mathsf{F}\,\psi'$ and $\widetilde{\mathsf{Y}}\,\psi'$ is the hypothetical previous time operator, which is true if either ψ' is true in the previous time-step or such a time-step does not exist. 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 [Gab87]), which allows to eliminate all pure-past formulas[2]. Note that all the above substitutions start from the innermost subformula. Item 4 proceeds as for the translation of pCTL* into CTL (see Lemma 3.3 and Theorem 3.4 of [KP95]). 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, ft are respectively pure-past, pure-present and pure-future formulas, we need to substitute *present* by $\mathfrak{false}$ in ps and ft and by $\mathfrak{true}$ in pr. For items 3 and 4 the non-elementary blow-up is inherited from the use of the Separation Theorem. Item 5 follows by using the formula $\varphi = \langle\langle A\rangle\rangle \mathsf{G}\,(\bigwedge_{i=1}^{n}(p_i \Leftrightarrow [\![\emptyset]\!] p_i) \Rightarrow (p_0 \Leftrightarrow [\![\emptyset]\!] p_0))$ (resp., $\varphi = \langle\langle A\rangle\rangle \mathsf{G}\,(\bigwedge_{i=1}^{n}(p_i \Leftrightarrow \mathsf{P}\,((\widetilde{\mathsf{Y}}\,\mathfrak{false}) \wedge p_i)) \Rightarrow (p_0 \Leftrightarrow \mathsf{P}\,((\widetilde{\mathsf{Y}}\,\mathfrak{false}) \wedge p_0))))$, which is similar to that used to prove that pLTL is exponentially more succinct than LTL (see Theorem 3.1 of [LMS02]). By using an argument similar to that used in [LMS02], we obtain the desired result. Item 6 follows by using a proof similar to that used for m¯CTL* (see Theorem 3.4 of [KV06]), and so showing that the ATL formula $\varphi = \langle\langle A\rangle\rangle \mathsf{F}\,(([\![\emptyset]\!] \mathsf{X}\, p) \wedge ([\![\emptyset]\!] \mathsf{X}\, \neg p))$ 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.

[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.

Corollary 3.5. mATL*, $\overline{\text{p}}$ATL*, pATL*, *and* mpATL* *have the same expressive power of* ATL*. $\overline{\text{m}}$ATL* *and* m$\overline{\text{p}}$ATL* *have the same expressive power, but are less expressive than* ATL*. *Moreover, all of them are at least exponentially more succinct than* ATL*.

Fig. 1 summarizes all the above results regarding expressiveness and succinctness. The acronym *"lin"* (resp., *"ne"*) means that the translation exists and it is linear (resp., nonelementarily) in the size of the formula, and *"/"* means that such a translation is impossible. The numbers in brackets represent the item of Theorem 3.4 in which the translation is shown. We use no numbers when the translation is trivial or comes by a composition of existing ones.

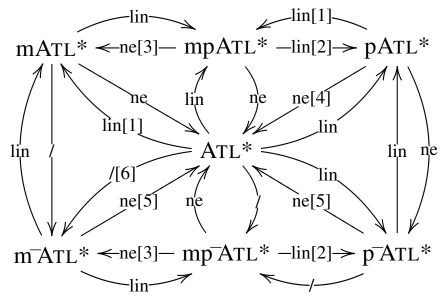

Figure 1: Hierarchy of expressive power and succinctness.

4 Decision Procedures

In this section, we study the satisfiability and model-checking problems for mpATL*. We directly study the richer mpATL* logic, since we prove the 2EXPTIME upper bound for this logic. To obtain such upper bounds, we use an automata-theoretic approach by introducing a novel automaton model: *agent-action tree automata with satellites*.

4.1 Agent-action tree automata with satellites

Alternating tree automata [MS87] 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* [JW95] 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 some of its successors or to all its successors. Hence, the automaton does not distinguish between directions. As a generalization of alternating automata (both in the symmetric and asymmetric cases), here we consider *agent-action tree automata* (AGCTA, for short), which can send copies to successor nodes, according to agents' decisions. These automata are a slight variation of *automata over concurrent game structures*, which were introduced in [SF06]. Moreover, we also consider AGCTA along with the satellite framework (AGCTAS, for short), in a similar way it has been done in [KV06]. 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 easily a tight complexity of the considered problems w.r.t. the size of the specification. We use symmetric AGCTAS for the satisfiability and asymmetric AGCTAS for the model-checking. In the following, we simply write AGCTA when we indifferently refer to its symmetric or asymmetric version. The formal definitions of AGCTA and AGCTAS follow.

Definition 4.2. *A* symmetric AGCTA *is a tuple* $\mathcal{A} = \langle \Sigma, \mathrm{Ag}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$*, where* Σ*,* Ag*, and* Q *are non-empty finite sets of* input symbols*,* agents*, and* states*, respectively,* $q_0 \in \mathrm{Q}$ *is an* initial state*,* F *is an* acceptance condition *to be defined later, and* $\delta : \mathrm{Q} \times \Sigma \mapsto \mathrm{B}^+(\mathrm{D} \times \mathrm{Q})$ *is an* alternating transition function*, where* $\mathrm{D} = \{\Diamond, \Box\} \times 2^{\mathrm{Ag}}$ *is an extended set of abstract directions, which maps a state and an input symbol to a positive boolean combination of two kinds of* atoms*:* existential atoms $((\Diamond, A), q')$ *and* universal atoms $((\Box, A), q')$*. Moreover,* $\mathcal{A}$ *is* asymmetric *if it also contains a set* Ac *of actions (i.e.,* $\mathcal{A} = \langle \Sigma, \mathrm{Ag},$

$\mathrm{Ac}, \mathrm{Q}, \delta, q_0, \mathrm{F}\rangle$) *and* $\delta : \mathrm{Q} \times \Sigma \mapsto \mathrm{B}^+(\mathrm{Ac}^{\mathrm{Ag}} \times \mathrm{Q})$ *contains atoms of the form* (d, q')*, where* d *is a decision of the agents in* Ag.

Definition 4.3. *A* run *of a symmetric* AGCTA $\mathcal{A}$ *on a* Σ*-labeled* AAT $\mathcal{T} = \langle \mathrm{Ag}, \mathrm{Ac}, \mathrm{T}, \mathsf{v} \rangle$ *is a* $(\mathrm{Q} \times \mathrm{T})$*-labeled* $\mathbb{N}$*-tree* $\mathcal{R} = \langle \mathrm{Tr}, \mathsf{r} \rangle$ *such that* (i) $\mathsf{r}(\varepsilon) = (q_0, \varepsilon)$ *and* (ii) *for all* $y \in \mathrm{Tr}$*, with* $\mathsf{r}(y) = (q, x)$*, there is a set* $\mathrm{S} \subseteq \mathrm{D} \times \mathrm{Q}$*, with* $\mathrm{S} \models \delta(q, \mathsf{v}(x))$*, such that for all atoms* $(a, q') \in \mathrm{S}$ *it holds that*

- *if* $a = (\Diamond, A)$ *then there exists a decision* $\mathsf{d}_A \in \mathrm{Ac}^A$ *such that for all counterdecisions* $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag} \setminus A}$ *it holds that* $(q', x \cdot (\mathsf{d}_A, \mathsf{d}^c_A)) \in \mathsf{L}(y)$*, where* $\mathsf{L}(y)$ *is the set* $\{\mathsf{r}(y \cdot y') \mid y' \in \mathbb{N}, y \cdot y' \in \mathrm{Tr}\}$ *of labels of successors of* y *in* $\mathcal{R}$*;*

- *if* $a = (\Box, A)$ *then for all decisions* $\mathsf{d}_A \in \mathrm{Ac}^A$ *there exists a counterdecision* $\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag} \setminus A}$ *such that* $(q', x \cdot (\mathsf{d}_A, \mathsf{d}^c_A)) \in \mathsf{L}(y)$.

If $\mathcal{A}$ *is asymmetric, then the above item* (ii) *is substituted by the following:* (ii') *for all* $y \in \mathrm{Tr}$*, with* $\mathsf{r}(y) = (q, x)$*, there exists a set* $\mathrm{S} \subseteq \mathrm{D} \times \mathrm{Q}$*, with* $\mathrm{S} \models \delta(q, \mathsf{v}(x))$*, such that for all atoms* $(\mathsf{d}, q') \in \mathrm{S}$ *it holds that* $(q', x \cdot \mathsf{d}) \in \mathsf{L}(y)$.

In this paper, we only consider automata along with a co-Büchi acceptance condition $\mathrm{F} \subseteq \mathrm{Q}$. A run $\mathcal{R}$ on a AAT $\mathcal{T}$ for an AGCTA $\mathcal{A}$ with a co-Büchi condition is accepting iff for all its paths all states in F only occur finitely often. A tree $\mathcal{T}$ is accepted by $\mathcal{A}$ iff there is an accepting run of $\mathcal{A}$ on it. By $\mathcal{L}(\mathcal{A})$ we denote the language accepted by the automaton $\mathcal{A}$, i.e., the set of all the AATs that $\mathcal{A}$ accepts. $\mathcal{A}$ is said *empty* if $\mathcal{L}(\mathcal{A}) = \emptyset$. The emptiness problem for $\mathcal{A}$ is to decide whether $\mathcal{L}(\mathcal{A}) = \emptyset$.

We now define AGCTA with satellite.

Definition 4.4. *An* asymmetric *(resp.,* symmetric*)* AGCTA with satellite *(*AGCTAS*) is a tuple* $\langle \mathcal{A}, \mathcal{D} \rangle$*, where* $\mathcal{A} = \langle \Sigma \times \mathrm{Q}', \mathrm{Ag}, \mathrm{Ac}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$ *(resp.,* $\mathcal{A} = \langle \Sigma \times \mathrm{Q}', \mathrm{Ag}, \mathrm{Q}, \delta, q_0, \mathrm{F} \rangle$*) is an* asymmetric *(resp.,* symmetric*)* AGCTA *and* $\mathcal{D}$ *is a* satellite $\langle \Sigma', \mathrm{Q}', \delta', q'_0 \rangle$*, where* $\Sigma \subseteq \Sigma'$ *and* Q' *are non-empty finite sets of* input symbols *and* states*,* $q'_0 \in \mathrm{Q}'$ *is an* initial state*, and* $\delta' : \mathrm{Q}' \times \Sigma' \mapsto \mathrm{Q}'$ *is a* deterministic transition function*.*

For the coming definition we need an extra notation. Let f be a Boolean formula, by $f[p/q]$ we denote the formula in which all occurrences of p in f are replaced by q.

Definition 4.5. *An* AAT $\mathcal{T}$ *is accepted by an* asymmetric *(resp.,* symmetric*)* AGCTAS $\langle \mathcal{A}, \mathcal{D} \rangle$ *iff* $\mathcal{T}$ *is accepted by the* AGCTA *product-automata* $\mathcal{A}^* = \langle \Sigma, \mathrm{Ag}, \mathrm{Ac}, \mathrm{Q} \times \mathrm{Q}', \delta^*, (q_0, q'_0), \mathrm{F}^* \rangle$ *(resp.,* $\mathcal{A}^* = \langle \Sigma, \mathrm{Ag}, \mathrm{Q} \times \mathrm{Q}', \delta^*, (q_0, q'_0), \mathrm{F}^* \rangle$*), where* F^* *is the acceptance condition directly derived from* F *and* δ^* *is such that:* $\delta^*((q, p), \sigma) = \delta(q, (\sigma, p))[q'/(q', \delta'(p, \sigma))]$*, for* $\sigma \in \Sigma$ *and* $(q, p) \in \mathrm{Q} \times \mathrm{Q}'$.

In words, $\delta^*((q, p), \sigma)$ is obtained by substituting in $\delta(q, (\sigma, p))$ each occurrence of a state q' with a tuple of the form (q', p'), where $p' = \delta'(p, \sigma)$ is the new state of the satellite. As for AGCTA, we consider AGCTAS along with a co-Büchi acceptance condition. W.r.t. Definition 4.5, we have that $\mathrm{F}^* = \mathrm{F} \times \mathrm{Q}'$. Moreover, we set $\mathcal{L}(\langle \mathcal{A}, \mathcal{U} \rangle) = \mathcal{L}(\mathcal{A}^*)$.

Note that satellites are just a convenient way to describe an AGCTA in which the state space can be partitioned into two components, one of which is deterministic and independent from the other, and has no influence on the acceptance. Indeed, it is just a matter of technicality to see that AGCTAS inherit all the closure properties of the alternating automata. In particular, the following theorem shows how the separation between $\mathcal{A}$ and $\mathcal{U}$ enables a tight analysis of the complexity of the relative emptiness problem.

Theorem 4.6. *The emptiness problem for a symmetric (resp., asymmetric) co-Büchi* AGCTAS $\langle \mathcal{A}, \mathcal{D} \rangle$*, where* $\mathcal{A}$ *has* m *agents and* n *states and* $\mathcal{D}$ *has* n' *states, can be decided in time* $2^{O((n \cdot \log(n \cdot n'))^m)}$.

Sketch. The proof proceeds as follow. First, we use the bounded model theorem for symmetric AGCTA (see Theorem 2 of [SF06]), which asserts that an AGCTA accepts an AAT iff it accepts a $|atom(\mathcal{A}) \times \mathrm{Ag}|^{|\mathrm{Ag}|}$-bounded AAT, in order to obtain a linear translation from the symmetric AGCTA $\mathcal{A}$ to an asymmetric one $\mathcal{A}'$ with the same sets of agents and states and $|atom(\mathcal{A}) \times \mathrm{Ag}|$ actions, such that $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$, and $\mathcal{L}(\mathcal{A}') = \emptyset$ iff $\mathcal{L}(\mathcal{A}) = \emptyset$. The transition function of $\mathcal{A}'$ is obtained from that of $\mathcal{A}$ by substituting each existential atom $((\Diamond, A), q')$ (resp., universal atom $((\Box, A), q')$) with the formula $\bigvee_{\mathsf{d}_A \in \mathrm{Ac}^A} \bigwedge_{\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag} \setminus A}} ((\mathsf{d}_A, \mathsf{d}^c_A), q')$ (resp., $\bigwedge_{\mathsf{d}_A \in \mathrm{Ac}^A} \bigvee_{\mathsf{d}^c_A \in \mathrm{Ac}^{\mathrm{Ag} \setminus A}} ((\mathsf{d}_A, \mathsf{d}^c_A), q')$. As second step, since $\mathcal{A}'$ can be seen as a classical alternating co-Büchi tree automaton with $(atom(\mathcal{A}) \times \mathrm{Ag})^{\mathrm{Ag}}$ as set of directions, we use an exponential-time translation that leads to an asymmetric nondeterministic Büchi AGCTA $\mathcal{A}''$ with the same sets of agents and actions and $2^{O(n \cdot \log(n))}$ states such that $\mathcal{L}(\mathcal{A}'') = \mathcal{L}(\mathcal{A})$ (see Theorem 1.2 of [MS95]). At this point, taking the product-automata between $\mathcal{A}''$ and the satellite $\mathcal{D}$ we obtain another asymmetric nondeterministic Büchi AGCTA $\mathcal{A}'''$ with $2^{O(n \cdot \log(n \cdot n'))}$ states such that $\mathcal{L}(\langle \mathcal{A}'', \mathcal{D} \rangle) = \mathcal{L}(\mathcal{A}''')$. Now, by construction, it is evident that $\mathcal{L}(\langle \mathcal{A}, \mathcal{D} \rangle) = \emptyset$ iff $\mathcal{L}(\mathcal{A}''') = \emptyset$. Finally, the emptiness of $\mathcal{A}'''$ can be checked in a quadratic running-time in the size of the transition function, which is polynomial in the number of states and exponential in the number of directions (see Theorem 2.2 of [VW86]). Overall, with this procedure, we obtain that the emptiness problem for symmetric (resp., asymmetric) co-Büchi AGCTAS is solveable in exponential time w.r.t. $n \cdot \log(n \cdot n')$ and double exponential in the number m of agents. Precisely, in time $2^{O((n \cdot \log(n \cdot n'))^m)}$. □

4.7 From path formulas to satellites

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 formulas with past operators (pLTL) [Gab87, LPZ85] can be translated into automata on infinite words of bounded size [Var88], and that pLTL 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 introduce the following notation. A basic formula b in φ is a subformula of φ of the form $b = \langle\!\langle A_b \rangle\!\rangle \psi_b$. Observe that the present trace for b is irrelevant, so we directly write $\mathcal{G}, \rho \models b$. By $sub(\varphi)$ we denote the set of all basic subformulas of φ and by $dsub(\varphi) \subseteq sub(\varphi)$ the immediate subformulas of φ. Finally, we use the following abbreviations $\mathrm{AP}_\varphi = \mathrm{AP} \cup dsub(\varphi)$, $\mathrm{AP}^*_\varphi = \mathrm{AP} \cup sub(\varphi)$, $\mathrm{AP}^{pr}_\varphi = \mathrm{AP}_\varphi \cup \{present\}$, and $\mathrm{AP}^{*,pr}_\varphi = \mathrm{AP}^*_\varphi \cup \{present\}$.

Before showing the full satellite construction, we first show 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 direct basic subformulas $b' \in dsub(b)$ by the label b' read as atomic proposition. By using a slight variation of the procedure developed in [Var88], we can translate $\widehat{\psi}_b$ into a universal co-Büchi word automaton[3] $\mathcal{U}_b = \langle 2^{\mathrm{AP}^{pr}_b}, \mathrm{Q}_b, \delta_b, \mathrm{Q}_{0b}, \mathrm{F}_b \rangle$, with a number of states at most exponential in $|\psi_b|$, that accepts all and only the infinite words on $2^{\mathrm{AP}^{pr}_b}$ that are models of $\widehat{\psi}_b$. By applying the classical subset construction to $\mathcal{U}_b$, we obtain the satellite $\mathcal{D}_b = \langle 2^{\mathrm{AP}^{pr}_b}, 2^{\mathrm{Q}_b}, \delta^d_b, \mathrm{Q}_{0b} \rangle$, where, for all sets $Q \subseteq \mathrm{Q}_b$ and labels $\sigma \subseteq \mathrm{AP}^{pr}_b$, it holds that $\delta^d_b(Q, \sigma) = \bigcup_{q \in Q} \delta_b(q, \sigma)$. To better understand the usefulness of the satellite $\mathcal{D}_b$, consider $\mathcal{U}_b$ after that a prefix w' of an infinite word $w \in (2^{\mathrm{AP}^{pr}_b})^\omega$ 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 of the word w. Consider now the satellite $\mathcal{D}_b$ after that the same prefix w' is read. Since $\mathcal{D}_b$ is deterministic, there is only one active state that, by construction, is 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 $\mathcal{U}_b$.

[3] Word automata can be seen as tree automata in which the tree has just one path. Moreover, a universal word automaton accepts a word iff all its runs are accepting.

We now define two different satellites, which we use for satisfiability and model-checking. Regarding satisfiability, we have to maintain, at the same time, a memory for all path formulas ψ_b contained in the mpATL* formula φ that we want to check. To this aim, we build the product-satellite $\mathcal{D}_\varphi = \langle 2^{\mathrm{AP}_\varphi^{*,pr}}, \prod_{b \in sub(\varphi)} 2^{\mathrm{Q}_b}, \delta_\varphi^d, \prod_{b \in sub(\varphi)} \{\mathrm{Q}_{0b}\} \rangle$ over all the satellites $\mathcal{D}_b$, with $b \in sub(\varphi)$, where, for all $Q_b \subseteq \mathrm{Q}_b$ and $\sigma \subseteq \mathrm{AP}_\varphi^{*,pr}$, it is set $\delta_\varphi^d(\prod_{b \in sub(\varphi)} Q_b, \sigma) = \prod_{b \in sub(\varphi)} \{\delta_b^d(Q_b, \sigma \cap \mathrm{AP}_b^{pr})\}$. Regarding model-checking, since we verify one basic formulas $b \in sub(\varphi)$ at a time, we build the product-satellite $\mathcal{D}_{b,P}^* = \langle 2^{\mathrm{AP}_b^{*,pr}}, \mathrm{Q}_b^*, \delta_b^*, P \rangle$ over all the satellites $\mathcal{D}_{b'}$, with $b' \in sub(b)$, where, for all $Q_{b'} \subseteq \mathrm{Q}_{b'}$ and $\sigma \subseteq \mathrm{AP}_b^{*,pr}$, it is set $\mathrm{Q}_b^* = \prod_{b' \in sub(b)} 2^{\mathrm{Q}_{b'}}$, $P \in \mathrm{Q}_b^*$, and $\delta_b^*(\prod_{b' \in sub(b)} Q_{b'}, \sigma) = \prod_{b' \in sub(b)} \{\delta_{b'}^d(Q_{b'}, \sigma \cap \mathrm{AP}_{b'}^{pr})\}$. Note that the size of the satellites $\mathcal{D}_\varphi$ and $\mathcal{D}_{b,P}^*$, i.e., the number of their states, is bounded by $2^{O(2^{|\varphi|})}$ and $2^{O(2^{|\psi_b|})}$, respectively.

4.8 Satisfiability

The satisfiability procedure we now propose technically extends that used for ATL* in [Sch08] along with that for mCTL* in [KV06]. 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 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 [Sch08]). The main difference of our definition of explicit models w.r.t. that given in [Sch08] 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 AGCTAS that accepts all and only the explicit models of the specification. The proof follows that used in Theorem 4 of [Sch08] and changes w.r.t. the use of the satellite $\mathcal{D}_\varphi$ that helps the main automaton $\mathcal{A}$ whenever it needs to start with the verification of a given path formula ψ_b, with $b \in sub(\varphi)$. In particular, $\mathcal{A}$ needs to send to the successors of a node x labeled with b in the AAT 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 [KV06], this requirement is satisfied by $\mathcal{A}$ by defining the transition function, for the part of interest, as follows: $\delta(q_b, (\sigma, Q)) = ((\Box, \mathrm{Ag}), q_b) \wedge \bigwedge_{q \in Q_{|b}} \bigwedge_{q' \in \delta_b(q, \sigma \cap \mathrm{AP}_b \cup \{present\})} ((\Box, \mathrm{Ag}), (q', new))$, where $b \in \sigma$ and $Q_{|b}$ is the state of $\mathcal{D}_b$ in the product-state set Q. Putting the above reasoning all together, the following result holds.

Theorem 4.9. *Given a* mpATL* *formula* φ*, we can build a symmetric co-Büchi* AGCTAS $\langle \mathcal{A}, \mathcal{D}_\varphi \rangle$*, where* $\mathcal{D}_\varphi$ *has* $2^{O(2^{|\varphi|})}$ *states and* $\mathcal{A}$ *has* $O(2^{|\varphi|})$ *states and contains all and only the agents used in* φ*, such that* $\mathcal{L}(\langle \mathcal{A}, \mathcal{D}_\varphi \rangle)$ *is exactly the set of all the tree models of* φ*.*

Using Theorems 4.6 and 4.9, we obtain that the check of the existence of a model for a given mpATL* specification φ can be done in time $2^{2^{O(|\varphi|^2)}}$, resulting in a 2EXPTIME algorithm in the size of φ. Since mpATL* subsumes mCTL*, which has a satisfiability problem 2EXPTIME-HARD [KV06], we then derive the following result.

Theorem 4.10. *The satisfiability problem for* mpATL* *is* 2EXPTIME-COMPLETE*.*

4.11 Model checking

As for ATL*, for mpATL* we use a bottom-up model-checking algorithm. The procedure we propose extends that used for ATL* in [AHK02] by means of the satellite. Note that this procedure is different from that used for mCTL* in [KV06], which is top-down and uses a local model-checking method. I.e., it checks whether the initial state satisfies the formula. Contrarily, our procedure is a global model checking that returns all states satisfying the formula. We now give the main idea behind our procedure.

Consider a CGS $\mathcal{G}$ and an mpATL* formula φ. If one uses directly the procedure from [AHK02], each state s of $\mathcal{G}$ turns labeled by a basic subformula b of φ together with a possible initial trace ρ ending in s iff $\mathcal{G}, \rho \models b$. Then, one can check whether $\mathcal{G}, \rho \models b$ by building an AGCTA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) \neq \emptyset$ iff $\mathcal{G}, \rho \models b$. Usually, $\mathcal{A}$ is the product of two different automata $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A}_1$ is used to select, according to A_b agents' strategy, all subtrees coming from the unwinding of $\mathcal{G}$ starting at s and $\mathcal{A}_2$ is used to verify that such subtrees satisfy ψ_b with ρ being the present. Although this procedure seems reasonable, it cannot be used because of the fact that we have infinitely many possible initial traces, while the set of atomic proposition in a CGS, as well as the number of checks the procedure can perform, have to be finite. A solution we propose here is to substitute ρ with "finite information", which is supplied by a satellite. In particular observe that, to manage the memoryful quantification, we only need an amount of memory whose size just depends on the size of the formula. Indeed, suppose b is the innermost basic subformula of φ, it is possible to prove that, if we have two traces ρ_1 and ρ_2 with the same final state and such that the satellites $\mathcal{D}_b$ reading the two words related to ρ_1 and ρ_2 reach the same state, then $\mathcal{G}, \rho_1 \models b$ iff $\mathcal{G}, \rho_2 \models b$. Using this fact, we can substitute the trace of the above procedure, with the relative state of the satellite, thus partitioning the information carried by the traces into equivalence classes.

We now describe the complete model-checking procedure for mpATL*. We start with the innermost basic formulas b of φ and terminate with its direct basic subformulas. For the base case, we use an automaton similar to that used in the previous sketch. So, we can build an extended CGS $\mathcal{G}'$ such that each state s of $\mathcal{G}'$ is labeled by a pair (b, Q), with $Q \in 2^{\mathrm{Q}_b}$, iff $\mathcal{G}, \rho \models b$, for all the traces ρ ending in s such that, when $\mathcal{D}_b$ reads the word related to ρ it reaches the state Q. For the iterative case, assume that there is an extended CGS $\mathcal{G}_b$ for which the satisfaction of all the basic subformulas of b has already been determined. Then, using $\mathcal{G}_b$, we can build an AGCTAS $\mathcal{A}_{P,Q}$, with $\mathcal{D}^*_{b,P}$ as satellite, where $P \in \mathrm{Q}^*_b$ and $Q \in 2^{\mathrm{Q}_b}$, such that $\mathcal{L}(\mathcal{A}_{P,Q}) \neq \emptyset$ iff $\mathcal{G}, \rho \models b$, where the word related to ρ on the extended CGS $\mathcal{G}_b$ carries the satellite $\mathcal{D}^*_{b,P_0}$, with $P_0 = \prod_{b' \in sub(b)} 2^{\mathrm{Q}_{0b'}}$, to the state P and the satellite $\mathcal{D}_b$ to the state Q. As for the classical procedure, also the main automaton of $\mathcal{A}_{P,Q}$ is the product of two different automata. The first one selects, using the satellite and accordingly to A_b agents' strategy, all subtrees coming from the unwinding of $\mathcal{G}$ starting at s, which carry in their labeling also the atomic propositions related to the basic subformulas of b. The second one verifies that such subtrees satisfy ψ_b with P "being the present". The resulting automaton is then used to have an extended structure that also includes the satisfaction of the formula b. By applying the procedure recursively, we obtain an enriched model for each basic formula $b \in sub(\varphi)$. Hence, we can determine whether the input formula φ is satisfied by the original structure $\mathcal{G}$ or not (for more technical details, see the full version of the paper). By a simple calculation, it follows that the over all procedure takes time $|\mathcal{G}|^{2^{O(|\varphi|^2)}}$, 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 3.4, 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 [AHK02], we then derive the following result.

Theorem 4.12. *The model checking problem for* mpATL* *is* PTIME-COMPLETE *w.r.t. the size of the model and* 2EXPTIME-COMPLETE *w.r.t. the size of the specification.*

A Full definition of mpATL* syntax and semantics

The syntax of mpATL* is formally defined as follows.

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

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

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

The semantics of mpATL* is formally defined as follows.

Definition A.2. *Given a* CGS $\mathcal{G} = \langle \mathrm{AP}, \mathrm{Ag}, \mathrm{Ac}, \mathrm{St}, \lambda, \tau, s_0 \rangle$ *and two initial traces* $\rho, \rho_p \in \mathrm{Trc}$*, where* $\rho = \rho' \cdot s$*,* $\rho' \in \mathrm{Trc} \cup \{\varepsilon\}$*, and* $s \in \mathrm{St}$*, it holds that:*

1. $\mathcal{G}, \rho, \rho_p \models \mathsf{present}$ *iff* $\rho = \rho_p$*;*
2. $\mathcal{G}, \rho, \rho_p \models p$*, for* $p \in \mathrm{AP}$*, iff* $p \in \lambda(s)$*;*
3. $\mathcal{G}, \rho, \rho_p \models \neg\varphi$ *iff not* $\mathcal{G}, \rho, \rho_p \models \varphi$*, that is* $\mathcal{G}, \rho, \rho_p \not\models \varphi$*;*
4. $\mathcal{G}, \rho, \rho_p \models \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{G}, \rho, \rho_p \models \varphi_1$ *and* $\mathcal{G}, \rho, \rho_p \models \varphi_2$*;*
5. $\mathcal{G}, \rho, \rho_p \models \varphi_1 \vee \varphi_2$ *iff* $\mathcal{G}, \rho, \rho_p \models \varphi_1$ *or* $\mathcal{G}, \rho, \rho_p \models \varphi_2$*;*
6. $\mathcal{G}, \rho, \rho_p \models \langle\!\langle A \rangle\!\rangle \psi$ *iff there exists an s-defined strategy* f_A *of agents in A such that for all* f_A*-plays* π *it holds that* $\mathcal{G}, \rho' \cdot \pi, 0, \rho \models \psi$*;*
7. $\mathcal{G}, \rho, \rho_p \models [\![A]\!] \psi$ *iff for all the s-defined strategies* f_A *of agents in A there exists an* f_A*-play* π *such that* $\mathcal{G}, \rho' \cdot \pi, 0, \rho \models \psi$.

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

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

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

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

References

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *JACM*, 49(5):672–713, 2002.

[DTV00] M. Daniele, P. Traverso, and M.Y. Vardi. Strong Cyclic Planning Revisited. In *ECP'99*, pages 35–48, 2000.

[EH86] E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching Versus Linear Time. *JACM*, 33(1):151–178, 1986.

[FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, Cambridge, Mass., 1995.

[Gab87] D.M. Gabbay. The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In *TLS'87*, LNCS 398, pages 409–448. Springer-Verlag, 1987.

[Jam04] W. Jamroga. Strategic Planning Through Model Checking of ATL Formulae. In *ICAISC'04*, LNCS 3070, pages 879–884. Springer-Verlag, 2004.

[JW95] D. Janin and I. Walukiewicz. Automata for the Modal μ-Calculus and Related Results. In *MFCS'95*, LNCS 969, pages 552–562. Springer-Verlag, 1995.

[KP95] O. Kupferman and A. Pnueli. Once and For All. In *LICS'95*, pages 25–35. IEEE Computer Society, 1995.

[KV06] O. Kupferman and M.Y. Vardi. Memoryful Branching-Time Logic. In *LICS'06*, pages 265–274. IEEE Computer Society, 2006.

[KVW00] O. Kupferman, M.Y. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time Model Checking. *JACM*, 47(2):312–360, 2000.

[LMS02] F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal Logic with Forgettable Past. In *LICS'02*, pages 383–392. IEEE Computer Society, 2002.

[LPZ85] O. Lichtenstein, A. Pnueli, and L.D. Zuck. The Glory of the Past. In *LP'85*, pages 196–218, 1985.

[MS87] D.E. Muller and P.E. Schupp. Alternating Automata on Infinite Trees. *TCS*, 54(2-3):267–276, 1987.

[MS95] 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. *TCS*, 141:69–107, 1995.

[NPP08] P. Niebert, D. Peled, and A. Pnueli. Discriminative Model Checking. In *CAV'08*, LNCS 5123, pages 504–516. Springer-Verlag, 2008.

[OR94] M.J. Osborne and A. Rubinstein. *A course in game theory.* MIT Press, 1994.

[PV07] M. Pistore and M.Y. Vardi. The Planning Spectrum - One, Two, Three, Infinity. *JAIR'07*, 30:101–132, 2007.

[Sch08] S. Schewe. ATL* Satisfiability is 2ExpTime-Complete. In *ICALP'08*, LNCS 5126, pages 373–385. Springer-Verlag, 2008.

[SF06] S. Schewe and B. Finkbeiner. Satisfiability and Finite Model Property for the Alternating-Time μ-Calculus. In *CSL'06*, pages 591–605, 2006.

[Var88] M.Y. Vardi. A Temporal Fixpoint Calculus. In *POPL'88*, pages 250–259, 1988.

[vdHW02] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *AAMAS'02*, pages 1167–1174, 2002.

[VW86] M.Y. Vardi and P. Wolper. Automata-Theoretic Techniques for Modal Logics of Programs. *JCSS*, 32(2):183–221, 1986.

[Woo01] M.J. Woolridge. *Introduction to Multiagent Systems.* John Wiley & Sons, 2001.