Deciding ATL* satisfiability by tableaux

Amélie David

Deciding ATL* Satisfiability by Tableaux

Amélie David(✉)

Laboratoire IBISC, Université D'Évry Val-d'Essonne,
EA 4526 23 Bd de France, 91037 Évry Cedex, France
adavid@ibisc.univ-evry.fr

Abstract. We propose a tableau-based decision procedure for the full Alternating-time Temporal Logic ATL*. We extend our procedure for ATL+ in order to deal with nesting of temporal operators. As a side effect, we obtain a new and conceptually simple tableau method for CTL*. The worst case complexity of our procedure is 3EXPTIME, which is suboptimal compared to the 2EXPTIME complexity of the problem. However our method is human-readable and easily implementable. A web application and binaries for our procedure are available at http://atila.ibisc.univ-evry.fr/tableau_ATL_star/.

Keywords: Alternating-time temporal logic · ATL* · Automated theorem prover · Satisfiability · Tableaux

1 Introduction

The logic ATL* is the full version of the Alternating-time Temporal Logic introduced in [1] in order to describe open systems, that is systems that can interact with their environment. Thus ATL and ATL* are the multi-agent versions of the branching-time temporal logics CTL and CTL*. Indeed, in ATL and ATL* the environment is modelled by an extra agent e interfering with the system components (the remaining agents) who need to succeed their task no matter how e responds to their actions. ATL* is an important extension of ATL and ATL+ (an intermediate logic between ATL and ATL*) since it allows one to express useful properties such as fairness constraints. Such properties can be expressed only if nesting of temporal operators is possible, which is not the case in ATL and ATL+. It is worth noting that ATL+ only permits Boolean combination of unnested temporal operators.

The problem studied in this paper is about deciding the satisfiability of ATL* formulae. Models for ATL* formulae are directed graphs called *concurrent game structures* where transitions between two states depend on the chosen action of each agent. In general, there exists two ways for deciding the satisfiability: using automata, as done in [8] or using tableaux; as we do here. In this paper,

All proofs of lemmas, propositions and theorems, as well as complete examples can be found in the version with appendices at https://www.ibisc.univ-evry.fr/~adavid/fichiers/cade15 tableaux atl star long.pdf.

we propose the first tableau-based decision procedure for ATL*, as well as the first implementation of a decision procedure for ATL*, which is also the first implementation of a tableau-based decision procedure for ATL+. We extend our procedure for ATL+ [2] following the natural basic idea: separate present and future. However, this extension is not trivial since the separation into present and future is more subtle than for ATL+ and needs to keep track of path formulae so as to be able to check eventualities. We think that our tableau-based decision procedure for ATL* is easy to understand and therefore also provides a new tableau-based decision procedure for CTL* which is conceptually simple. We prove that our procedure runs in at most 3EXPTIME, which is suboptimal (the optimal worst case complexity has been proved to be 2EXPTIME in [8]). However, we do not know of any specific cases where our procedure runs in 3EXPTIME, which leaves the possibility that it is optimal, after all.

This paper is organized as follows: Sect. 2 gives the syntax and semantics for ATL*. The general structure of the tableau-based decision procedure for ATL* that we propose can be found in Sect. 3 and details of the procedure in Sects. 4, 5 and 6. Theorems about soundness, completeness and complexity are given in Sect. 7 with their sketch of proof. The Sect. 8 is about the implementation of our procedure. The paper ends with some concluding remarks indicating also some possible directions of future research.

2 Syntax and Semantics of ATL*

ATL* can be seen as an extension of the *computational tree logic* (CTL*) [5] where the path quantifiers E – there exists a path – and A – for all paths – are replaced by $\langle\langle \mathsf{A} \rangle\rangle$ and $[\![\mathsf{A}]\!]$ where A is a coalition of agents. Intuitively $\langle\langle \mathsf{A} \rangle\rangle \Phi$ means "There exists a strategy for the coalition A such that, no matter which strategy the remaining agents follow, Φ holds". On the other hand, $[\![\mathsf{A}]\!] \Phi$ means "For all strategies of the coalition A, there exists a strategy of the remaining agents such that Φ holds". Also, whereas *transition systems* or *Kripke structures* are used in order to evaluate CTL* formulae, *concurrent game models* (CGM), whose definition is given in the Sect. 2.2 are used to evaluate ATL* formulae.

2.1 Syntax of ATL*

Before giving the syntax of ATL*, we recall that, as for CTL* or LTL, $\bigcirc$, $\Box$ and $\mathcal{U}$ mean "*Next*", "*Always*" and "*Until*" respectively. In this paper, we give the syntax in negation normal form over a fixed set $\mathbb{P}$ of atomic propositions and primitive temporal operators $\bigcirc$ "*Next*", $\Box$ "*Always*" and $\mathcal{U}$ "*Until*". The syntax of ATL* in negation normal form is defined as follows:

$$\textit{State formulae: } \varphi := l \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \langle\langle \mathsf{A} \rangle\rangle \Phi \mid [\![\mathsf{A}]\!] \Phi \tag{1}$$

$$\textit{Path formulae: } \Phi := \varphi \mid \bigcirc \Phi \mid \Box \Phi \mid (\Phi \mathcal{U} \Phi) \mid (\Phi \vee \Phi) \mid (\Phi \wedge \Phi) \tag{2}$$

where $l \in \mathbb{P} \cup \{\neg p \mid p \in \mathbb{P}\}$ is a literal, $\mathbb{A}$ is a fixed finite set of agents and $\mathsf{A} \subseteq \mathbb{A}$ is a coalition. Note that $\top := p \vee \neg p$, $\bot := \neg \top$, $\neg \langle\langle \mathsf{A} \rangle\rangle \Phi := [\![\mathsf{A}]\!] \neg \Phi$.

The temporal operator "*Sometimes*" $\Diamond$ can be defined as $\Diamond\varphi := \top\,\mathcal{U}\varphi$ and the temporal operator "*Release*" as $\psi\,\mathcal{R}\varphi := \Box\varphi \vee \varphi\,\mathcal{U}(\varphi \wedge \psi)$. When unnecessary, parentheses can be omitted.

In this paper, we use φ, ψ, η to denote arbitrary state formulae and Φ, Ψ to denote path formulae. By an ATL^* formula we will mean by default a *state* formula of ATL^*.

2.2 Concurrent Game Models

As for ATL or ATL^+, ATL^* formulae are evaluated over *concurrent game models*. Concurrent games models are transition systems where each transition to a unique successor state results from the combination of actions chosen by all the agents (components and/or environment) of the system.

Notation: Given a set X, $\mathcal{P}(X)$ denotes the power set of X.

Definition 1 (Concurrent game model and structure). *A concurrent game model (in short CGM) is a tuple* $\mathcal{M} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out}, \mathbb{P}, \mathsf{L})$ *where*

- $\mathbb{A} = \{1, \ldots, k\}$ *is a finite, non-empty set of* players (agents),
- $\mathbb{S}$ *is a non-empty set of* states,
- *for each agent* $\mathsf{a} \in \mathbb{A}$, Act_a *is a non-empty set of actions. For any coalition* $\mathsf{A} \subseteq \mathbb{A}$ *we denote* $\mathsf{Act}_\mathsf{A} := \prod_{\mathsf{a}\in A} \mathsf{Act}_\mathsf{a}$ *and use* σ_A *to denote a tuple from* Act_A. *In particular,* $\mathsf{Act}_\mathbb{A}$ *is the set of all possible* action vectors *in* $\mathcal{M}$.
- *for each agent* $\mathsf{a} \in \mathbb{A}$, $\mathsf{act}_\mathsf{a} : \mathbb{S} \to \mathcal{P}(\mathsf{Act}_\mathsf{a}) \setminus \{\emptyset\}$ *defines for each state* s *the actions available to* a *at* s,
- out *is a transition function assigning to every state* $s \in \mathbb{S}$ *and every action vector* $\sigma_\mathbb{A} = \{\sigma_1, \ldots, \sigma_k\} \in \mathsf{Act}_\mathbb{A}$ *a state* $\mathsf{out}(s, \sigma_\mathbb{A}) \in \mathbb{S}$ *that results from* s *if every agent* $\mathsf{a} \in \mathbb{A}$ *plays action* σ_a, *where* $\sigma_\mathsf{a} \in \mathsf{act}_\mathsf{a}(s)$ *for every* $\mathsf{a} \in \mathbb{A}$.
- $\mathbb{P}$ *is a non-empty set of atomic propositions.*
- $\mathsf{L} : \mathbb{S} \to \mathcal{P}(\mathbb{P})$ *is a labelling function.*

The sub-tuple $\mathcal{S} = (\mathbb{A}, \mathbb{S}, \{\mathsf{Act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \{\mathsf{act}_\mathsf{a}\}_{\mathsf{a}\in\mathbb{A}}, \mathsf{out})$ *is called a* concurrent game structure *(CGS).*

2.3 Semantics of ATL*

In order to give the semantics of ATL^*, we use the following notions. Although they are the same as those in [2], we recall them here to make the paper self-contained.

Computations. A *play*, or *computation*, is an infinite sequence $s_0 s_1 s_2 \cdots \in \mathbb{S}^\omega$ of states such that for each $i \geq 0$ there exists an action vector $\sigma_\mathbb{A} = \langle \sigma_1, \ldots, \sigma_k \rangle$ such that $\mathsf{out}(s_i, \sigma_\mathbb{A}) = s_{i+1}$. A *history* is a finite prefix of a play. We denote by $\mathsf{Plays}_\mathcal{M}$ and $\mathsf{Hist}_\mathcal{M}$ respectively the set of plays and set of histories in $\mathcal{M}$.

For a state $s \in \mathbb{S}$, we use $\mathsf{Plays}_{\mathcal{M}}(s)$ and $\mathsf{Hist}_{\mathcal{M}}(s)$ as the set of plays and set of histories with initial state s. Given a sequence of states λ, we denote by λ_0 its initial state, by λ_i its $(i+1)$th state, by $\lambda_{\leq i}$ the prefix $\lambda_0 \dots \lambda_i$ of λ and by $\lambda_{\geq i}$ the suffix $\lambda_i \lambda_{i+1} \dots$ of λ. When $\lambda = \lambda_0 \dots \lambda_\ell$ is finite, we say that it has length ℓ and write $|\lambda| = \ell$. Also, we set $last(\lambda) = \lambda_\ell$.

Strategies. A *strategy* for an agent a in $\mathcal{M}$ is a mapping $F_{\mathsf{a}} : \mathsf{Hist}_{\mathcal{M}} \to \mathsf{Act}_{\mathsf{a}}$ such that for all histories $h \in \mathsf{Hist}_{\mathcal{M}}$, we have $F_{\mathsf{a}}(h) \in \mathsf{act}_{\mathsf{a}}(last(h))$. This kind of strategies is also known as "perfect recall" strategies. We denote by $\mathsf{Strat}_{\mathcal{M}}(\mathsf{a})$ the set of strategies of agent a. A collective strategy of a coalition $\mathsf{A} \subseteq \mathbb{A}$ is a tuple $(F_{\mathsf{a}})_{\mathsf{a} \in \mathsf{A}}$ of strategies, one for each agent in A. We denote by $\mathsf{Strat}_{\mathcal{M}}(\mathsf{A})$ the set of collective strategies of coalition A. A play $\lambda \in \mathsf{Plays}_{\mathcal{M}}$ is consistent with a collective strategy $F_{\mathsf{A}} \in \mathsf{Strat}_{\mathcal{M}}(\mathsf{A})$ if for every $i \geq 0$ there exists an action vector $\sigma_{\mathbb{A}} = \langle \sigma_1, \dots, \sigma_k \rangle$ such that $\mathsf{out}(\lambda_i, \sigma_{\mathbb{A}}) = \lambda_{i+1}$ and $\sigma_{\mathsf{a}} = F_{\mathsf{a}}(\lambda_{\leq i})$ for all $\mathsf{a} \in \mathsf{A}$. The set of plays with initial state s that are consistent with F_{A} is denoted $\mathsf{Plays}_{\mathcal{M}}(s, F_{\mathsf{A}})$. For any coalition $\mathsf{A} \subseteq \mathbb{A}$ and a given state $s \in \mathbb{S}$ in a given CGM $\mathcal{M}$, an A *-co-action* at s in $\mathcal{M}$ is a mapping $\mathsf{Act^c}_{\mathsf{A}} : \mathsf{Act}_{\mathsf{A}} \to \mathsf{Act}_{\mathbb{A} \setminus \mathsf{A}}$ that assigns to every collective action of A at the state s a collective action at s for the complementary coalition $\mathbb{A} \setminus \mathsf{A}$. Likewise, an A *-co-strategy* in $\mathcal{M}$ is a mapping $F^c_{\mathsf{A}} : \mathsf{Strat}_{\mathcal{M}}(\mathsf{A}) \times \mathsf{Hist}_{\mathcal{M}} \to \mathsf{Act}_{\mathbb{A} \setminus \mathsf{A}}$ that assigns to every collective strategy of A and every history h a collective action at $last(h)$ for $\mathbb{A} \setminus \mathsf{A}$, and $\mathsf{Plays}_{\mathcal{M}}(s, F^c_{\mathsf{A}})$ is the set of plays with initial state s that are consistent with F^c_{A}.

Semantics. The semantics of ATL^* is the same as the one of CTL^* [5] (modulo CGM as intended interpretations) with the exception of the two following items:

- $\mathcal{M}, s \models \langle\!\langle \mathsf{A} \rangle\!\rangle \Phi$ iff there exists an A-strategy F_{A} such that, for all computations $\lambda \in \mathsf{Plays}_{\mathcal{M}}(s, F_{\mathsf{A}})$, $\mathcal{M}, \lambda \models \Phi$
- $\mathcal{M}, s \models [\![\mathsf{A}]\!] \Phi$ iff there exists an A-co-strategy F^c_{A} such that, for all computations $\lambda \in \mathsf{Plays}_{\mathcal{M}}(s, F^c_{\mathsf{A}})$, $\mathcal{M}, \lambda \models \Phi$

Valid, satisfiable and equivalent formulae in ATL^* are defined as usual.

3 Tableau-Based Decision Procedure for ATL*

In this section, we give the general description of our tableau-based decision procedure for ATL^* formulae. The different steps of the procedure are summarized in this section and Fig. 1 and then detailed in the next three sections.

From an initial formula η, the tableau-based decision procedure for ATL^* that we propose attempts to build step-by-step a directed graph from which it is possible to extract a CGM for η. This attempt will lead to a failure if η is not satisfiable.

Nodes of that graph are labelled by sets of *state formulae* and are partitioned into two categories: *prestates* and *states*.

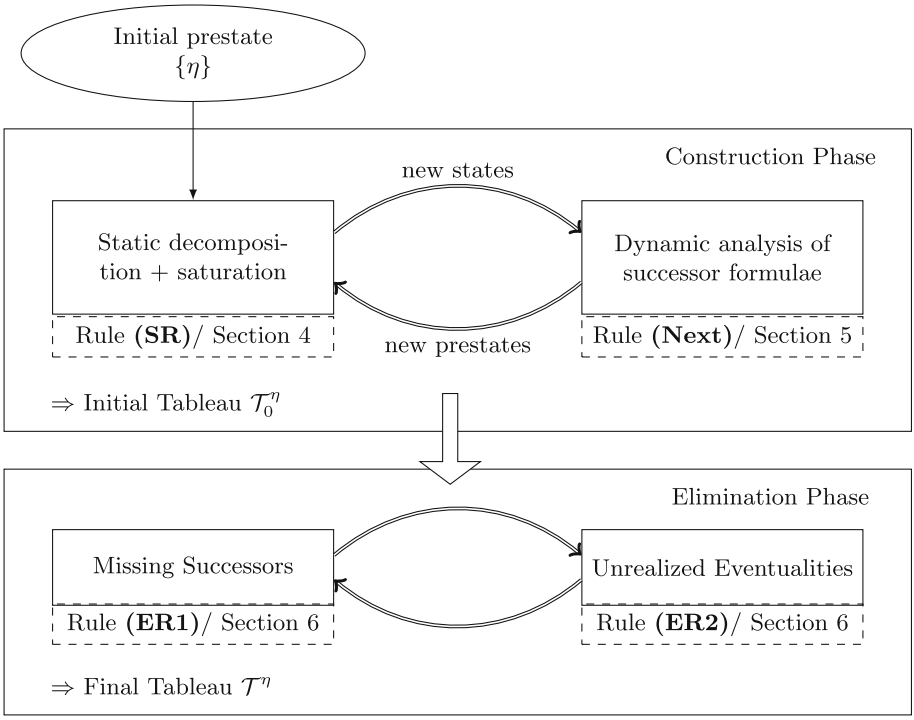

Fig. 1. Overview of the tableau-based decision procedure for ATL*

A prestate can be seen as a node where the information contained in its formulae is "implicit". When we decompose all the formulae of a prestate and saturate the prestate, we obtain one or several states as successor nodes. States have the particularity of containing formulae of the form $\langle\langle \mathsf{A} \rangle\rangle \bigcirc \varphi$ or $[\![\mathsf{A}]\!] \bigcirc \varphi$ from which it is possible to compute the next steps of the tableau creation. All prestates have states as successors and directed edges between them are of the form $\Longrightarrow$; on the other hand, all states have prestates as successors and directed edges between them are of the form $\xrightarrow{\sigma_{\mathbb{A}}}$ where $\sigma_{\mathbb{A}}$ is an action vector.

The procedure is in two phases: the *construction phase* and the *elimination phase*. First, we create an initial node, that is a prestate containing the initial formula η, and we construct the graph by expanding prestates into states via a rule called **(SR)** and by computing prestates from states with a rule called **(Next)**. The rule **(SR)** decomposes each ATL* formula of a prestate, and then saturates the prestate into new states. Explanation of rules **(SR)** and **(Next)** can be found in Sects. 4 and 5, respectively.

The procedure avoids creation of duplicated nodes (a form of loop check), which ensures termination of the procedure. The construction phase ends when no new states can be added to the graph. The graph obtained at the end of the construction phase is called the *initial tableau for* η, also noted $\mathcal{T}_0^\eta$.

The second phase of the procedure eliminates via the rule **(ER1)** all nodes with missing successors, that is prestates with no more successors at all or states with at least one missing action vector on its outcome edges. Also, by means of a rule called **(ER2)** it eliminates all states with "unrealized eventualities", that is states that cannot ensure that all the objectives it contains will be eventually fulfilled. The graph obtained at the end of the elimination phase of the procedure is called the *final tableau for* η, also noted $\mathcal{T}^{\eta}$. Explanation of rules **(ER1)** and **(ER2)** can be found in Sect. 6.

Our tableau-based decision procedure for ATL^* deals with what [6] calls "tight satisfiability": the set $\mathbb{A}$ of agents involved in the tableau (and the CGMs it tries to build) is the set of agents present in the input formula.

4 Construction Phase: Decomposition and Saturation

Decomposition of ATL^* Formulae All ATL^* formulae can be partitioned into four categories: *primitive formulae*, α*-formulae*, β*-formulae* and γ*-formulae*. Primitive formulae correspond to the "simplest formulae" in the sense that they cannot be decomposed. These formulae are $\top, \bot$, the literals and all ATL^* *successor formulae*, of the form $\langle\langle \mathsf{A} \rangle\rangle \bigcirc \psi$ or $[\![\mathsf{A}]\!] \bigcirc \psi$ where ψ is called the *successor component* of $\langle\langle \mathsf{A} \rangle\rangle \bigcirc \psi$ or $[\![\mathsf{A}]\!] \bigcirc \psi$ respectively. Every non-primitive formula must be decomposed into primitive formulae. α-formulae are of the form $\varphi \wedge \psi$ where φ and ψ are α-components while β-formulae are of the form $\varphi \vee \psi$ where φ and ψ are β-components. Their decomposition is classical. Other formulae, that is of the form $\langle\langle \mathsf{A} \rangle\rangle \Phi$ or $[\![\mathsf{A}]\!] \Phi$, where $\Phi \neq \bigcirc \psi$, are γ-formulae. This notion firstly introduced in [2] reveals quite useful also in the more expressive context of ATL^*. Decomposition of these formulae is trickier than for α- and β-formulae. Indeed, we will need to extract all possibilities of truth encapsulated in γ-formula ξ, which concretely aims at defining one or several conjunctions of primitive formulae such that their disjunction is equivalent to the γ-formulae ξ (see lemma 1).

Decomposition of γ-Formulae. This subsection contains the heart of the decision procedure for ATL^*, indeed the main difference with our decision procedure for ATL^+ lies in the treatment of γ-formulae. The first difficulty is that quantifiers $\langle\langle \mathsf{A} \rangle\rangle$ or $[\![\mathsf{A}]\!]$ cannot distribute over Boolean connectors as seen in [2]. An additional difficulty specific to ATL^* is the fact that it is now necessary to also deal with nesting of temporal operators, resulting in a second level of recurrence when the temporal operators $\Box$ and $\mathcal{U}$ are encountered in the decomposition function described below.

In temporal logics, e.g. LTL, the operator $\mathcal{U}$ is considered as an eventuality operator, that is an operator that promises to verify a given formula at some instant/state. When we write $\lambda \models \varphi_1 \mathcal{U} \varphi_2$, where φ_1 and φ_2 are state formulae, we mean that there is a state λ_i of the computation λ where φ_2 holds and φ_1 holds for all the states of λ preceding λ_i. So, once the property φ_2 is verified, we do not need to take care of φ_1, φ_2 and $\varphi_1 \mathcal{U} \varphi_2$ any more. We say that $\varphi_1 \mathcal{U} \varphi_2$ is *realized*. However, if φ_1 and φ_2 are path formulae, e.g. $\Box \Phi_1$ and $\Box \Phi_2$ respectively, state λ_i is such that from it Φ_2 must hold forever – we say that $\Box \Phi_2$

is "initiated" at λ_i, in the sense that we start to make $\Box\Phi_2$ true at λ_i –, and for every computation $\lambda_{\geq j}$, where $j < i$, $\Box\Phi_1$ must hold. So Φ_1 has to be true forever, that is even after $\Box\Phi_2$ had been initiated. This explains the fact that at a possible state s the path formula $\varphi_1 \mathcal{U} \varphi_2$ may become $\varphi_1 \mathcal{U} \varphi_2 \wedge \varphi_1$ when φ_1 is a path formula and we postpone φ_2. Note that φ_1 is then also initiated at s. We now face the problem of memorizing the fact that a path formula Φ is initiated since path formulae cannot be stored directly in a state. That is why, during the decomposition of γ-formulae, we add a new set of path formulae linked to a γ-component and the current state.

The definition and general treatment of eventualities in our procedure are given in Sect. 6.

In order to decompose γ-formulae $\varphi = \langle\langle \mathsf{A} \rangle\rangle \Phi$ or $\varphi = [\![\mathsf{A}]\!] \Phi$, we analyse the path formula Φ in terms of present (current state) and future (next states). This analysis is done by a *γ-decomposition* function $\mathsf{dec} : \mathsf{ATL}^*_\mathsf{p} \rightarrow \mathcal{P}(\mathsf{ATL}^*_\mathsf{s} \times \mathsf{ATL}^*_\mathsf{p} \times \mathcal{P}(\mathsf{ATL}^*_\mathsf{p}))$ where $\mathsf{ATL}^*_\mathsf{p}$ is the set of ATL^* path formulae and $\mathsf{ATL}^*_\mathsf{s}$ is the set of ATL^* state formulae. Intuitively, the function dec assigns to the path formula Φ, a set of triples $\langle \psi, \Psi, S \rangle$ where ψ is a state formula true at the current state, Ψ is a path formula expressing what must be true at next states and S is the set of path formulae initiated at the current state during the γ-decomposition This set S will be used during the elimination phase to determine if eventualities are realized or not, see Sect. 6.

We first define two operators $\otimes$ and $\oplus$ between two sets $\mathfrak{S}_1$ and $\mathfrak{S}_2$ of triples.

- $\mathfrak{S}_1 \otimes \mathfrak{S}_2 := \{\langle \psi_i \mathbin{\dot\wedge} \psi_j, \Psi_i \mathbin{\dot\wedge} \Psi_j, S_i \cup S_j \rangle \mid \langle \psi_i, \Psi_i, S_i \rangle \in \mathfrak{S}_1, \langle \psi_j, \Psi_j, S_j \rangle \in \mathfrak{S}_2\}$
- $\mathfrak{S}_1 \oplus \mathfrak{S}_2 := \{\langle \psi_i \mathbin{\dot\wedge} \psi_j, \Psi_i \mathbin{\dot\vee} \Psi_j, S_i \cup S_j \rangle \mid \langle \psi_i, \Psi_i, S_i \rangle \in \mathfrak{S}_1,\ \langle \psi_j, \Psi_j, S_j \rangle \in \mathfrak{S}_2, \Psi_i \neq \top, \Psi_j \neq \top\}$

The function dec is defined by induction on the structure of path formula Φ as follows:

- $\mathsf{dec}(\varphi) = \{\langle \varphi, \top, \emptyset \rangle\}$ for any ATL^* state formula φ
- $\mathsf{dec}(\bigcirc \Phi_1) = \{\langle \top, \Phi_1, \emptyset \rangle\}$ for any path formula Φ_1
- $\mathsf{dec}(\Box \Phi_1) = \{\langle \top, \Box\Phi_1, \{\Phi_1\} \rangle\} \otimes \mathsf{dec}(\Phi_1)$
- $\mathsf{dec}(\Phi_1 \mathcal{U} \Phi_2) = (\{\langle \top, \Phi_1 \mathcal{U} \Phi_2, \{\Phi_1\} \rangle\} \otimes \mathsf{dec}(\Phi_1)) \cup (\{\langle \top, \top, \{\Phi_2\} \rangle\} \otimes \mathsf{dec}(\Phi_2))$
- $\mathsf{dec}(\Phi_1 \wedge \Phi_2) = \mathsf{dec}(\Phi_1) \otimes \mathsf{dec}(\Phi_2)$
- $\mathsf{dec}(\Phi_1 \vee \Phi_2) = \mathsf{dec}(\Phi_1) \cup \mathsf{dec}(\Phi_2) \cup (\mathsf{dec}(\Phi_1) \oplus \mathsf{dec}(\Phi_2))$

Note that the definition of the function dec is based on the fixed-point equivalences of LTL [4]: $\Box\Psi \equiv \Psi \wedge \bigcirc\Box\Psi$ and $\Phi \mathcal{U} \Psi \equiv \Psi \vee (\Phi \wedge \bigcirc(\Phi \mathcal{U} \Psi))$.

The operators $\dot\wedge$ and $\dot\vee$ correspond respectively to the operators $\wedge$ and $\vee$ where the associativity, commutativity, idempotence and identity element properties are embedded in the syntax. The aim of both $\dot\wedge$ and $\dot\vee$ is to automatically transform resultant formulae in conjunctive normal form without redundancy, and therefore ensures the termination of our tableau-based decision procedure. For instance, when applying the function dec on $\Box\Diamond\Phi \wedge \Diamond\Phi$ we may obtain a path formula $\Box\Diamond\Phi \wedge \Diamond\Phi \wedge \Diamond\Phi$ and applying again the function dec on the so-obtained path formula will return $\Box\Diamond\Phi \wedge \Diamond\Phi \wedge \Diamond\Phi \wedge \Diamond\Phi$, and so on forever. Also when the

formula is complicated with $\wedge$ and $\vee$ embedded in temporal operators, we may not be able to define which part of a path formula is identical to another one. We avoid these unwanted behaviours with our use of $\dot{\wedge}$ and $\dot{\vee}$ and the transformation of any new path formula in conjunctive normal form without redundancies.

Now, let $\zeta = \langle\langle \mathsf{A} \rangle\rangle \Phi$ or $\zeta = [\![\mathsf{A}]\!]\Phi$ be a γ-formula to be decomposed. Each triple $\langle \psi, \Psi, S \rangle \in \mathsf{dec}(\Phi)$ is then converted to a *γ-component* $\gamma_c(\psi, \Psi, S)$ as follows:

$$\gamma_c(\psi, \Psi, S) = \psi \quad \text{if } \Psi = \top \tag{3}$$

$$\gamma_c(\psi, \Psi, S) = \psi \wedge \langle\langle \mathsf{A} \rangle\rangle \bigcirc \langle\langle \mathsf{A} \rangle\rangle \Psi \quad \text{if } \zeta \text{ is of the form } \langle\langle \mathsf{A} \rangle\rangle \Phi, \tag{4}$$

$$\gamma_c(\psi, \Psi, S) = \psi \wedge [\![\mathsf{A}]\!] \bigcirc [\![\mathsf{A}]\!] \Psi \quad \text{if } \zeta \text{ is of the form } [\![\mathsf{A}]\!]\Phi \tag{5}$$

and a γ-set $\gamma_s(\psi, \Psi, S) = S$.

The following key lemma claims that every γ-formula is equivalent to the disjunction of its γ-components.

Lemma 1. *For any* ATL^* *γ-formula* $\zeta = \langle\langle \mathsf{A} \rangle\rangle \Phi$ *or* $\zeta = [\![\mathsf{A}]\!]\Phi$

1. $\Phi \equiv \bigvee \{ \psi \wedge \bigcirc \Psi \mid \langle \psi, \Psi, S \rangle \in \mathsf{dec}(\Phi) \}$
2. $\langle\langle \mathsf{A} \rangle\rangle \Phi \equiv \bigvee \{ \langle\langle \mathsf{A} \rangle\rangle (\psi \wedge \bigcirc \Psi) \mid \langle \psi, \Psi, S \rangle \in \mathsf{dec}(\Phi) \}$, *and*
   $[\![\mathsf{A}]\!]\Phi \equiv \bigvee \{ [\![\mathsf{A}]\!] (\psi \wedge \bigcirc \Psi) \mid \langle \psi, \Psi, S \rangle \in \mathsf{dec}(\Phi) \}$
3. $\langle\langle \mathsf{A} \rangle\rangle \Phi \equiv \bigvee \{ \gamma_c(\psi, \Psi, S) \mid \langle \psi, \Psi, S \rangle \in \mathsf{dec}(\Phi) \}$

Example 1. (Decomposition of $\theta = \langle\langle 1 \rangle\rangle ((\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r))$*).* First, we apply the decomposition function to the path formula $\Phi = (\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r)$, see Fig. 2. We recall that $\Diamond\varphi \equiv \top \mathcal{U} \varphi$. It is worth noting that p and r can be replaced by any state formulae without affecting the basic structure of the computation of the function dec.

Then, for instance, from the triple $\langle r, \Box\Diamond q \wedge \Diamond q, \{r, \Diamond q\} \rangle$ of $\mathsf{dec}(\Phi)$, we obtain the γ-component $\gamma_c(r, \Box\Diamond q \wedge \Diamond q, \{r, \Diamond q\}) = r \wedge \langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle (\Box\Diamond q \wedge \Diamond q)$ and the γ-set $\gamma_s(r, \Box\Diamond q \wedge \Diamond q, \{r, \Diamond q\}) = \{r, \Diamond q\}$; from the triple $\langle \top, \Box\Diamond q \wedge \Diamond q, \{\Diamond q\} \rangle$ we obtain $\gamma_c(\top, \Box\Diamond q \wedge \Diamond q, \{\Diamond q\}) = \langle\langle 1 \rangle\rangle \bigcirc \langle\langle 1 \rangle\rangle (\Box\Diamond q \wedge \Diamond q)$ and $\gamma_s(\top, \Box\Diamond q \wedge \Diamond q, \{\Diamond q\}) = \{\Diamond q\}$.

Closure. The closure $cl(\varphi)$ of an ATL^* state formula φ is the least set of ATL^* formulae such that $\varphi, \top, \bot \in cl(\varphi)$, and $cl(\varphi)$ is closed under taking successor, α-, β- and γ-components of φ. For any set of state formulae Γ we define

$$cl(\Gamma) = \bigcup \{ cl(\psi) \mid \psi \in \Gamma \} \tag{6}$$

We denote by $|\psi|$ the length of ψ and by $||\Gamma||$ the cardinality of Γ.

Lemma 2. *For any* ATL^* *state formula* φ, $||cl(\varphi)|| < 2^{2^{2|\varphi|}}$.

Sketch of proof. The double exponent, in the size of the φ, of the closure comes from the fact that, during decomposition of γ-formulae, path formulae are put in disjunctive normal form. We recall that this form is necessary to ensure the termination of our procedure.

$$\mathsf{dec}(\Phi) = \mathsf{dec}(\Box\Diamond q \vee \Diamond r) \otimes \mathsf{dec}(\Diamond q \vee \Diamond r)$$

$$\begin{aligned}
\textcircled{1}\ \mathsf{dec}(\Box\Diamond q \vee \Diamond r) &= \mathsf{dec}(\Box\Diamond q) \cup \mathsf{dec}(\Diamond r) \cup (\mathsf{dec}(\Box\Diamond q) \oplus \mathsf{dec}(\Diamond r))\\
\quad \textbf{❶}\ \mathsf{dec}(\Box\Diamond q) &= \{\langle \top, \Box\Diamond q, \{q\}\rangle\} \otimes \quad (\{\langle \top, \Diamond q, \emptyset\rangle\} \otimes \mathsf{dec}(\top) \cup\\
&\qquad\qquad\qquad\qquad \{\langle \top, \top, \{q\}\rangle\} \otimes \mathsf{dec}(q))\\
&= \{\langle \top, \Box\Diamond q, \{\Diamond q\}\rangle\} \otimes \{\langle \top, \Diamond q, \emptyset\rangle, \langle q, \top, \{q\}\rangle\}\\
&= \{\langle \top, \Box\Diamond q \mathbin{\dot\wedge} \Diamond q, \{\top, \Diamond q\}\rangle, \langle q, \Box\Diamond q, \{\Diamond q, q\}\rangle\}\\
\quad \textbf{❷}\ \mathsf{dec}(\Diamond q) &= \{\langle \top, \Diamond r, \emptyset\rangle, \langle r, \top, \{r\}\rangle\}\\
\textcircled{1}\ \mathsf{dec}(\Box\Diamond q \vee \Diamond r) &= \{\langle \top, \Box\Diamond q \wedge \Diamond q, \{\Diamond q\}\rangle, \langle q, \Box\Diamond q, \{\Diamond q, q\}\rangle,\\
&\quad \langle \top, \Diamond r, \emptyset\rangle, \langle r, \top, \{r\}\rangle, \langle q, \Box\Diamond q \vee \Diamond r, \{\Diamond q, q\}\rangle,\\
&\quad \langle \top, (\Box\Diamond q \mathbin{\dot\wedge} \Diamond q) \mathbin{\dot\vee} \Diamond r \equiv (\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r), \{\Diamond q\}\rangle,\\
&\quad \}\\
\textcircled{2}\ \mathsf{dec}(\Diamond q \vee \Diamond r) &= \mathsf{dec}(\Diamond q) \cup \mathsf{dec}(\Diamond r) \cup (\mathsf{dec}(\Diamond q) \oplus \mathsf{dec}(\Diamond r))\\
&= \{\langle \top, \Diamond q, \emptyset\rangle, \langle q, \top, \{q\}\rangle\} \ \cup \{\langle \top, \Diamond r, \emptyset\rangle, \langle r, \top, \{r\}\rangle\}\\
&\quad \cup\{\langle \top, \Diamond q \vee \Diamond r, \emptyset\rangle\}\\
&= \{\langle \top, \Diamond q, \emptyset\rangle, \langle q, \top, \{q\}\rangle, \langle \top, \Diamond r, \emptyset\rangle, \langle r, \top, \{r\}\rangle,\\
&\quad \langle \top, \Diamond q \vee \Diamond r, \emptyset\rangle
\end{aligned}$$

$$\begin{aligned}
\mathsf{dec}(\Phi) = \textcircled{1} \otimes \textcircled{2} = \{&\langle \top, \Box\Diamond q \mathbin{\dot\wedge} \Diamond q \mathbin{\dot\wedge} \Diamond q \equiv \Box\Diamond q \wedge \Diamond q, \{\Diamond q\}\rangle, \dots\ ,\\
&\langle \top, (\Box\Diamond q \mathbin{\dot\vee} \Diamond r) \mathbin{\dot\wedge} (\Diamond q \mathbin{\dot\vee} \Diamond r) \mathbin{\dot\wedge} \Diamond q, \{\Diamond q\}\rangle, \dots\\
&\langle r, \Box\Diamond q \wedge \Diamond q, \{r, \Diamond q\}\rangle, \langle \top, \Box\Diamond q \wedge \Diamond q, \{\Diamond q\}\rangle, \dots\}
\end{aligned}$$

Fig. 2. Function dec applied on the path formula $\langle\langle 1\rangle\rangle((\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r))$

Full expansions of sets of ATL* formulae. Once we are able to decompose into components every non-primitive ATL* state formulae, it is possible to obtain full expansions of a given set of ATL* state formulae using the following definition:

Definition 2. *Let Γ, Δ be sets of ATL* state formulae and $\Gamma \subseteq \Delta \subseteq cl(\Gamma)$.*

1. *Δ is* patently inconsistent *if it contains $\bot$ or a pair of formulae φ and $\neg\varphi$.*
2. *Δ is a* full expansion of *Γ if it is not patently inconsistent and satisfies the following closure conditions:*
   - *if $\varphi \wedge \psi \in \Delta$ then $\varphi \in \Delta$ and $\psi \in \Delta$;*
   - *if $\varphi \vee \psi \in \Delta$ then $\varphi \in \Delta$ or $\psi \in \Delta$;*
   - *if $\varphi \in \Delta$ is a γ-formula, then at least one γ-component of φ is in Δ and exactly one of these γ-components, say $\gamma_c(\psi, \Psi, S)$, in Δ, denoted $\gamma_l(\varphi, \Delta)$, is designated as* the γ-component in Δ linked to the γ-formula *φ, as explained below. We also denote by $\gamma_{sl}(\varphi, \Delta)$ the set of path formulae $\gamma_s(\psi, \Psi, S)$, which is linked to the γ-component $\gamma_l(\varphi, \Delta)$*

The set of all full expansions of Γ is denoted by $\mathsf{FE}(\Gamma)$.

Proposition 1. *For any finite set of ATL* state formulae Γ:*

$$\bigwedge \Gamma \equiv \bigvee \left\{ \bigwedge \Delta \mid \Delta \in \mathsf{FE}(\Gamma) \right\}.$$

The proof easily follows from Lemma 1.

The rule **(SR)** adds to the tableau the set of full expansions of a prestate Γ as successor states of Γ.

Rule (SR). Given a prestate Γ, do the following:

1. For each full expansion Δ of Γ add to the pretableau a state with label Δ.
2. For each of the added states Δ, if Δ does not contain any formula of the form $\langle\langle \mathsf{A} \rangle\rangle \bigcirc \varphi$ or $[\![\mathsf{A}]\!] \bigcirc \varphi$, add the formula $\langle\langle \mathbb{A} \rangle\rangle \bigcirc \top$ to it;
3. For each state Δ obtained at steps 1 and 2, link Γ to Δ via a $\Longrightarrow$ edge;
4. If, however, the pretableau already contains a state Δ' with label Δ, do not create another copy of it but only link Γ to Δ' via a $\Longrightarrow$ edge.

5 Construction Phase: Dynamic Analysis of Successor Formulae

We recall that the considered agents are those explicitly mentioned in the initial formula η. The rule **(Next)** creates successor prestates to a given state, say Δ, so that the satisfiability of Δ is equivalent to the satisfiability of all the prestates. In our tableau construction procedure, choosing one of the successor formulae contained in Δ is considered as a possible action for every agent. Then each possible action vector is given a set of formulae corresponding to the choice collectively made by every agent. More details about the rationale behind the rule **(Next)** can be found in [3,6]. Moreover, it is worthwhile noticing that the rule **(Next)** is done in such a way so that any created prestate contains at most one formula of the form $[\![\mathsf{A}']\!] \bigcirc \psi$, where $\mathsf{A}' \neq \mathbb{A}$.

Rule (Next). Given a state Δ, do the following, where σ is a shorthand for $\sigma_{\mathbb{A}}$:

1. List all primitive successor formulae of Δ in such a way that all successor formulae of the form $\langle\langle \mathsf{A} \rangle\rangle \Phi$ precede all formulae of the form $[\![\mathsf{A}']\!] \Phi$ where $\mathsf{A}' \neq \mathbb{A}$, which themselves precede all formulae of the form $[\![\mathbb{A}]\!] \Phi$; let the result be the list

$$\mathbb{L} = \langle\langle \mathsf{A}_0 \rangle\rangle \bigcirc \varphi_0, \ldots, \langle\langle \mathsf{A}_{m-1} \rangle\rangle \bigcirc \varphi_{m-1}, \\ [\![\mathsf{A}'_0]\!] \bigcirc \psi_0, \ldots, [\![\mathsf{A}'_{l-1}]\!] \bigcirc \psi_{l-1}, [\![\mathbb{A}]\!] \bigcirc \mu_0, \ldots, [\![\mathbb{A}]\!] \bigcirc \mu_{n-1}$$

   Let $r_\Delta = \max\{m + l, 1\}$; we denote by $D(\Delta)$ the set $\{0, \ldots, r_\Delta - 1\}^{|\mathbb{A}|}$. Then, for every $\sigma \in D(\Delta)$, denote $N(\sigma) := \{i \mid \sigma_i \geqslant m\}$, where σ_i is the ith component of the tuple σ, and let $\mathsf{co}(\sigma) := [\sum_{i \in N(\sigma)} (\sigma_i - m)] \mod l$.
2. For each $\sigma \in D(\Delta)$ create a prestate:

$$\Gamma_\sigma = \{\varphi_p \mid \langle\langle \mathsf{A}_p \rangle\rangle \bigcirc \varphi_p \in \Delta \text{ and } \sigma_{\mathsf{a}} = p \text{ for all } \mathsf{a} \in \mathsf{A}_p\} \\ \cup \{\psi_q \mid [\![\mathsf{A}'_q]\!] \bigcirc \psi_q \in \Delta, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} - \mathsf{A}'_q \subseteq N(\sigma)\} \\ \cup \{\mu_r \mid [\![\mathbb{A}]\!] \bigcirc \mu_r \in \Delta\}$$

   If Γ_σ is empty, add $\top$ to it. Then connect Δ to Γ_σ with $\xrightarrow{\sigma}$.
   If, however, $\Gamma_\sigma = \Gamma$ for some prestate Γ that has already been added to the initial tableau, only connect Δ to Γ with $\xrightarrow{\sigma}$.

Example 2. We suppose a state containing the following successor formulae, that we arrange in the following way, where the first line of numbers corresponds to positions among negative successor formulae, and the second line corresponds to positions among successor formulae, with $\mathsf{A} \neq \mathbb{A}$.

$$\mathbb{L} = \overset{0}{\langle\!\langle 1 \rangle\!\rangle}(\Box\Diamond q \wedge \Diamond q), [\![1]\!] \bigcirc \overset{0}{\underset{1}{[\![1]\!]}}\Box\neg q, [\![2]\!] \bigcirc \overset{1}{\underset{2}{[\![2]\!]}}\Box\Diamond s, [\![1,2]\!] \bigcirc \neg q$$

The application of the rule **(Next)** on $\mathbb{L}$ gives the following results:

σ	$N(\sigma)$	$\mathsf{co}(\sigma)$	$\Gamma(\sigma)$	σ	$N(\sigma)$	$\mathsf{co}(\sigma)$	$\Gamma(\sigma)$
$0,0$	$\emptyset$	0	$\langle\!\langle 1 \rangle\!\rangle(\Box\Diamond q \wedge \Diamond q), \neg q$	$1,2$	$\{1,2\}$	1	$[\![1]\!]\Box\neg q, \neg q$
$0,1$	$\{2\}$	0	$\langle\!\langle 1 \rangle\!\rangle(\Box\Diamond q \wedge \Diamond q), [\![2]\!]\Box\Diamond s, \neg q$	$2,0$	$\{1\}$	1	$[\![1]\!]\Box\neg q, \neg q$
$0,2$	$\{2\}$	1	$\langle\!\langle 1 \rangle\!\rangle(\Box\Diamond q \wedge \Diamond q), \neg q$	$2,1$	$\{1,2\}$	1	$[\![1]\!]\Box\neg q, \neg q$
$1,0$	$\{1\}$	0	$\neg q$	$2,2$	$\{1,2\}$	0	$[\![2]\!]\Box\Diamond s, \neg q$
$1,1$	$\{1,2\}$	0	$[\![2]\!]\Box\Diamond s, \neg q$				

6 Elimination Phase

The *elimination phase* also works step-by-step. In order to go through one step to another we apply by turns two elimination rules, called **(ER1)** and **(ER2)**, until no more nodes can be eliminated. The rule **(ER1)** detects and deletes nodes with missing successor, while the rule **(ER2)** detects and delete states that do not realize all their eventualities. At each step, we obtain a new intermediate tableau, denoted by $\mathcal{T}_n^\eta$. We denote by S_n^η the set of nodes (states and prestates) of the intermediate tableau $\mathcal{T}_n^\eta$.

At the end of the elimination phase, we obtain the *final tableau* for η, denoted by $\mathcal{T}^\eta$. It is declared *open* if the initial node belongs to S^η, otherwise *closed*. The procedure for deciding satisfiability of η returns "No" if $\mathcal{T}^\eta$ is closed, "Yes" otherwise.

Remark 1. Contrary to the tableau-based decision procedure for ATL^+, we do not eliminate all the prestates at the beginning of the elimination phase. We eliminate them with the rule **(ER1)** only if necessary. This does not have any effect on the result of the procedure, nor any relevant modification in the soundness and completeness proofs, but it makes implementation quicker and easier.

Rule (ER1). Let $\Xi \in S_n^\eta$ be a node (prestate or state).

- In the case where Ξ is a prestate: if all nodes Δ with $\Xi \Longrightarrow \Delta$ have been eliminated at earlier stages, then obtain $\mathcal{T}_{n+1}^\eta$ by eliminating Ξ from $\mathcal{T}_n^\eta$.
- In the case where Ξ is a state: if, for some $\sigma \in D(\Xi)$, the node Γ with $\Xi \xrightarrow{\sigma} \Gamma$ has been eliminated at earlier stage, then obtain $\mathcal{T}_{n+1}^\eta$ by eliminating Ξ from $\mathcal{T}_n^\eta$.

In order to define the rule **(ER2)**, we first need to define what is an eventuality in the context of ATL^* and then define how to check whether eventualities are realized or not.

Eventualities. In our context, we consider all γ-formulae as *potential eventualities*. We recall that a γ-formula is of the form $\langle\langle \mathsf{A} \rangle\rangle \Phi$ or $[\![\mathsf{A}]\!]\Phi$ where $\Phi \neq \bigcirc\varphi$. When constructing a tableau step-by-step as we do in our procedure, it is possible to postpone forever promises encapsulated in operators such as $\mathcal{U}$ as far as we keep promising to satisfy them. We consider that a promise, which is a path formula, is satisfied (or realized) once it is initiated at the current state, which corresponds to an engagement to keep it true if necessary, for any computation starting at that state. So we want to know at a given state and for a given formula whether all promises (or eventualities) are realized. This is the role of the function $\mathsf{Realized}\colon \mathsf{ATL}^*_\mathsf{p} \times \mathcal{P}(\mathsf{ATL}^*_\mathsf{s}) \times \mathcal{P}(\mathsf{ATL}^*_\mathsf{p}) \to \mathbb{B}$, where $\mathbb{B}$ is the set $\{true, false\}$. The first argument of the function $\mathsf{Realized}$ is the path formula to study, the second argument is a set of state formulae Θ, and the third argument is a set of path formulae on which one is "engaged". This third argument is exactly what is added with respect to ATL^+ treatment. For our purpose, to know whether a potential eventuality is realized, we use the set Θ to represent the state containing the γ-formula and the set $S = \gamma_{sl}(\Phi, \Theta)$ obtained during the decomposition of Φ and the full expansion of Θ. This last set S is computed in Sect. 4 and corresponds to the set of path formulae initiated in the current state Θ. The definition of $\mathsf{Realized}$ is given by recursion on the structure of Φ as follows:

- $\mathsf{Realized}(\varphi, \Theta, S) = true$ iff $\varphi \in \Theta$
- $\mathsf{Realized}(\Phi_1 \wedge \Phi_2, \Theta, S) = \mathsf{Realized}(\Phi_1, \Theta, S) \wedge \mathsf{Realized}(\Phi_2, \Theta, S)$
- $\mathsf{Realized}(\Phi_1 \vee \Phi_2, \Theta, S) = \mathsf{Realized}(\Phi_1, \Theta, S) \vee \mathsf{Realized}(\Phi_2, \Theta, S)$
- $\mathsf{Realized}(\bigcirc\Phi_1, \Theta, S) = true$
- $\mathsf{Realized}(\Box\Phi_1, \Theta, S) = true$ iff $\Phi_1 \in \Theta \cup S$
- $\mathsf{Realized}(\Phi_2 \,\mathcal{U}\, \Phi_1, \Theta, S) = true$ iff $\Phi_1 \in \Theta \cup S$

Remark 2. In the two last items, we use the set $\Theta \cup S$ to handle the particular case where Φ_1 is a state formula that is already in the set Θ because of the behaviour of another coalition of agents.

We will see with Definition 4 that if the function $\mathsf{Realized}$ declares that an eventuality is not immediately realized at a given state, then we check in the corresponding successor states whether it is realized or not. But, because of the way γ-formulae are decomposed, an eventuality may change its form from one state to another. Therefore, we define the notion of *Descendant potential eventuality* in order to define a parent/child link between potential eventualities and keep track of not yet realized eventualities, and finally check whether the potential eventualities are realized at a given moment.

Definition 3. (Descendant potential eventualities). *Let Δ be a state and let $\xi \in \Delta$ be a potential eventuality of the form $\langle\langle \mathsf{A} \rangle\rangle \Phi$ or $[\![\mathsf{A}]\!]\Phi$. Suppose the γ-component $\gamma_l(\xi, \Delta)$ in Δ linked to ξ is, respectively, of the form $\psi \wedge \langle\langle \mathsf{A} \rangle\rangle \bigcirc \langle\langle \mathsf{A} \rangle\rangle \Psi$ or $\psi \wedge [\![\mathsf{A}]\!] \bigcirc [\![\mathsf{A}]\!] \Psi$. Then the* successor potential eventuality *of ξ w.r.t. $\gamma_l(\xi, \Delta)$ is the γ-formula $\langle\langle \mathsf{A} \rangle\rangle \Psi$ (resp. $[\![\mathsf{A}]\!]\Psi$) and it will be denoted by ξ^1_Δ. The notion of* descendant potential eventuality of ξ of degree d, *for $d > 1$, is defined inductively as follows:*

– *any successor eventuality of ξ (w.r.t. some γ-component of ξ) is a descendant eventuality of ξ of degree 1;*
– *any successor eventuality of a descendant eventuality ξ^n of ξ of degree n is a descendant eventuality of ξ of degree $n+1$.*

We will also consider ξ to be a descendant eventuality of itself of degree 0.

Realization of Potential Eventualities First, we give some notation:
Notation: Let $\mathbb{L} = \langle\langle \mathsf{A}_0\rangle\rangle \bigcirc \varphi_0, \ldots, \langle\langle \mathsf{A}_{m-1}\rangle\rangle \bigcirc \varphi_{m-1}, [\![\mathsf{A}'_0]\!] \bigcirc \psi_0, \ldots, [\![\mathsf{A}'_{l-1}]\!] \bigcirc \psi_{l-1}, [\![\mathbb{A}]\!] \bigcirc \mu_0, \ldots, [\![\mathbb{A}]\!] \bigcirc \mu_{n-1}$ be the list of all primitive successor formulae of $\Delta \in S_0^\eta$, induced as part of application of **(Next)**.
$Succ(\Delta, \langle\langle \mathsf{A}_p\rangle\rangle \bigcirc \varphi_p) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \sigma_{\mathsf{a}} = p \text{ for every } \mathsf{a} \in \mathsf{A}_p\}$
$Succ(\Delta, [\![\mathsf{A}'_q]\!] \bigcirc \psi_q) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma, \mathsf{co}(\sigma) = q \text{ and } \mathbb{A} - \mathsf{A}'_q \subseteq N(\sigma)\}$
$Succ(\Delta, [\![\mathbb{A}]\!] \bigcirc \mu_r) := \{\Gamma \mid \Delta \xrightarrow{\sigma} \Gamma\}$

Definition 4. (Realization of potential eventualities) *Let $\Delta \in S_n^\eta$ be a state and $\xi \in \Delta$ be a potential eventuality of the form $\langle\langle \mathsf{A}\rangle\rangle \Phi$ or $[\![\mathsf{A}]\!]\Phi$. Let $S = \gamma_{sl}(\xi, \Delta)$. Then:*

1. *If $\mathsf{Realized}(\Phi, \Delta, S) = true$ then ξ is realized at Δ in $\mathcal{T}_n^\eta$.*
2. *Else, let ξ^1_Δ be the successor potential eventuality of ξ w.r.t. $\gamma_l(\xi, \Delta)$. If for every $\Gamma \in Succ(\Delta, \langle\langle \mathsf{A}\rangle\rangle \bigcirc \xi^1_\Delta)$ (resp. $\Gamma \in Succ(\Delta, [\![\mathsf{A}]\!] \bigcirc \xi^1_\Delta)$), there exists $\Delta' \in \mathcal{T}_n^\eta$ with $\Gamma \Longrightarrow \Delta'$ and ξ^1_Δ is realized at Δ' in $\mathcal{T}_n^\eta$, then ξ is realized at Δ in $\mathcal{T}_n^\eta$.*

Example 3. Let $\Delta = \{\langle\langle 1\rangle\rangle((\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r)), [\![1]\!]\Box\neg q, [\![2]\!]\Box\Diamond s, [\![1,2]\!] \bigcirc \neg q, \neg q, \langle\langle 1\rangle\rangle \bigcirc \langle\langle 1\rangle\rangle(\Box\Diamond q \wedge \Diamond q), [\![1]\!] \bigcirc [\![1]\!]\Box\neg q, s, [\![2]\!] \bigcirc [\![2]\!]\Box\Diamond s\}$ be a state.
If we consider the potential eventuality $\xi = \langle\langle 1\rangle\rangle((\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r)) \in \Delta$, $\Phi = (\Box\Diamond q \vee \Diamond r) \wedge (\Diamond q \vee \Diamond r)$ and $S = \gamma_{sl}(\xi, \Delta) = \{\Diamond q\}$, then we obtain the following result:

$$\begin{aligned}\mathsf{Realized}(\Phi, \Delta, S) &= \mathsf{Realized}(\Box\Diamond q \vee \Diamond r, \Delta, S) \wedge \mathsf{Realized}(\Diamond q \vee \Diamond r, \Delta, S)\\ &= \mathsf{Realized}(\Box\Diamond q, \Delta, S) \vee \mathsf{Realized}(\Diamond r, \Delta, S) \wedge\\ &\quad\ \mathsf{Realized}(\Diamond q, \Delta, S) \vee \mathsf{Realized}(\Diamond r, \Delta, S)\\ &= (true \vee false) \wedge (false \vee false) = false\end{aligned}$$

The call of the function $\mathsf{Realized}$ on (Φ, Δ, S) returns $false$, which means that the potential eventuality ξ is not immediately realized. Therefore, we must check in the future if ξ can be realized or not. Concretely, we must check that the descendant potential eventuality $\xi^1 = \langle\langle 1\rangle\rangle(\Box\Diamond q \wedge \Diamond q)$ is realized at the next states corresponding to the collective choices of all agents to satisfy the successor formula $\langle\langle 1\rangle\rangle \bigcirc \xi^1$, that is states resulting from the transitions $(0,0)$, $(0,1)$ and $(0,2)$, as seen in Example 2.

Rule (ER2). If $\Delta \in S_n^\eta$ is a state and contains a potential eventuality that is not realized at $\Delta \in \mathcal{T}_n^\eta$, then obtain $\mathcal{T}_{n+1}^\eta$ by removing Δ from S_n^η.

7 Results and Sketches of Proofs

Theorem 1. *The tableau-based procedure for* ATL* *is sound with respect to unsatisfiability, that is if a formula is satisfiable then its final tableau is open.*

To prove soundness, we first prove that from any satisfiable prestate we obtain at least one satisfiable state, and we prove that from any satisfiable state we obtain only satisfiable prestates. Second, we prove that no satisfiable prestate or state can be eliminated via rule **(ER1)** or **(ER2)**, and in particular, if the initial prestate is satisfiable, it cannot be removed, which means that the tableau is open.

Theorem 2. *The tableau-based procedure for* ATL* *is complete with respect to unsatisfiability, that is if a tableau for an input formula is open then this formula is satisfiable.*

To prove completeness, we construct step-by-step a special structure called Hintikka structure from the open tableau and then we prove that a CGM satisfying the initial formula can be obtained from that Hintikka structure.

Theorem 3. *The tableau-based procedure for* ATL* *runs in at most* 3EXPTIME.

We first argue that the number of formulae in the closure of the initial formula η is at most double exponential in the size of η (see Lemma 2). Then we have that the number of states is at most exponential in the size of the closure of η. Therefore the procedure runs in at most 3EXPTIME.

8 Implementation of the Procedure

We propose a prototype implementing our tableau-based decision procedure for ATL*, available on the following web site: http://atila.ibisc.univ-evry.fr/tableau_ATL_star/.
This prototype aims at giving a user-friendly tool to the reader interested in checking satisfiability of ATL* formulae. This is why we provide our prototype as a web application directly ready to be used. The application allows one to enter a formula, or to select one from a predefined list of formulae, and then launch the computation of the corresponding tableau. It returns some statistics about the number of prestates and states generated as well as the initial and final tableaux for the input formula, therefore also an answer on its satisfiability. Explanation on how to use the application is given on the web site.

Our prototype is developed in Ocaml for the computation, and in PHP and JavaScript for the web interface. Binaries of the application can be found on the same web page.

As the main difference between ATL and ATL* comes from path formulae, we mainly focus our test on that point and use the list of tests proposed by Reynolds for CTL* in [7]. This allows us to check that our application gives the same results

in term of satisfiability and that our running times for these examples are satisfactory. Moreover, other tests using formulae with non trivial coalitions have been done. Nevertheless a serious benchmark has still to be done, which is a non trivial work, left for the future. Also, we plan to compare theoretically and experimentally our approach with the automata-decision based procedure of [8].

9 Conclusion

In this paper, we propose the first sound, complete and terminating tableau-based decision procedure for ATL^*: it is easy to understand and conceptually simple. We also provide the first implementation to decide the satisfiability of ATL^* formulae, among which ATL^+ formulae. In future works, it would be worthwhile to implement the automata-based decision procedure proposed in [8] and be able to make some practical comparisons. Another perspective is to implement model synthesis with a minimal number of states for satisfiable ATL^* formulae.

Acknowledgement. I would like to thank Serenella Cerrito and Valentin Goranko for their advices and proofreading. I also thank the anonymous referees for their helpful criticisms.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. Cerrito, S., David, A., Goranko, V.: Optimal tableaux-based decision procedure for testing satisfiability in the alternating-time temporal logic ATL+. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 277–291. Springer, Heidelberg (2014). http://arxiv.org/abs/1407.4645
3. Carral, D., Feier, C., Cuenca Grau, B., Hitzler, P., Horrocks, I.: $\mathcal{EL}$-ifying ontologies. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) IJCAR 2014. LNCS, vol. 8562, pp. 464–479. Springer, Heidelberg (2014)
4. Emerson, E.A.: Temporal and modal logics. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 995–1072. MIT Press (1990)
5. Emerson, E.A., Sistla, A.P.: Deciding full branching time logic. Inf. Control **61**(3), 175–201 (1984)
6. Goranko, V., Shkatov, D.: Tableau-based decision procedures for logics of strategic ability in multiagent systems. ACM Trans. Comput. Log. **11**(1), 1–48 (2009)
7. Reynolds, M.: A faster tableau for CTL. In: Puppis, G., Villa, T. (eds.) Proceedings Fourth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2013, Borca di Cadore, Dolomites, Italy, 29–31th August 2013. EPTCS, vol. 119, pp. 50–63 (2013). http://dx.doi.org/10.4204/EPTCS.119.7
8. Schewe, S.: ATL* satisfiability Is 2EXPTIME-complete. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 373–385. Springer, Heidelberg (2008)