From ATL Tableaux to Alternating Automata (Extended Draft)

Amélie David and Serenella Cerrito

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

Abstract. The logic called ATL (Alternating-time Temporal Logic) was introduced, in 1997, as a modal logic of agents. The first direct algorithm to test whether a given ATL-formula $\theta$ is satisfiable was provided by V. Goranko and G. van Drimmelen in 2006. Their algorithm is based on alternating infinite tree automata. In 2009, V. Goranko and D. Shkatov proposed a method to test satisfiability of a formula $\theta$ based on tableaux. In this work, we compare the two methods, and, building on this, we propose a new way of associating a specific kind of alternating automaton, the so called joker automaton, to a formula in order to test its satisfiability. The specificity of its construction is that it exploits the core of the tableau building procedure, and its interesting property is that the recognized language is the set of all models of $\theta$, without any restricting hypothesis on the moves of agents.

Keywords: Automata, Automated deduction, Logics for multi-agent systems, Modal logics, Tableaux, Temporal logics

1 Introduction

Alternating-time Temporal Logic — in short, ATL — belongs both to the family of temporal modal logics and to the class of modal logics for multiagent systems. Actually, CTL (Computational Tree Logic, [5]) can be seen as a special case of ATL with just one agent. CTL (or LTL [8]) is appropriate to model and verify closed reactive systems, which behavior is totally determined by the system states. ATL has been introduced in [3] as a logic suited to reason on open systems, i.e. systems whose environment may not be known in advance, so as that their behavior depends on both internal choices and environment choices. In few words, the essential difference between ATL and CTL is that ATL takes into account several agents, that can cooperate in order to achieve a goal. Thus, in an ATL language, path quantifiers are qualified by coalitions and express the existence of a coalition strategy capable to assure a goal no matter how the other agents play.

In the general case of temporal logics, the question of the relation between tableau based and automata based approaches to decide satisfiability is a natural one. However, up to our knowledge, this issue has been poorly investigated
in the case of ATL, where most of the studies focus on model checking. The issue of deciding ATL satisfiability has been studied, however, first in [7], then in [6]. These works propose two independent methods based, respectively, on alternating automata and tableaux. The alternating automata used in [7] take as inputs infinite trees having labels on nodes (but not on edges). These trees have a constant branching factor and can be seen as obtained by unwinding interpretations where each agent has the same number of moves, at each state, whereas models constructed by tableaux in [6] do not rely on such a restrictive hypothesis.

Here we study the relation between the two approaches, highlighting similarities and differences. Building on this, we propose a new way of associating a specific kind of alternating tree automaton, the so called joker automaton, to a formula $\theta$, in order to test its satisfiability. The specificity of its construction is that it exploits the core of the tableau building procedure, and its interesting property is that the language it recognizes is the set of all models of any given ATL-formula $\theta$, without no restricting hypothesis on the moves of agents. This property makes it interesting to both test $\theta$’s satisfiability and to model check whether any given concurrent game structure $\mathcal{M}$ is or not a model of $\theta$. Actually, we show that the essential of ATL tableau construction can also be seen as building an alternating automaton on infinite trees recognizing the whole class of models of the considered formula, exactly as it is the case for LTL with respect to Wolper’s tableaux [10] and Büchi automata [1].

The outline of the paper is the following. In Section 2 we give an account of ATL’s syntax and semantics. In Section 3 we recall and compare the methods for deciding ATL satisfiability proposed in [7, 6]. Section 4 is the core of the paper: it describes how to build a so called joker automaton out of a formula $\theta$ by exploiting tableau construction, and how to code any CGS interpreting $\theta$ as an input readable by such an automaton. Soundness and completeness properties holding for this automaton are stated and proved in Section 5. The paper ends with some concluding remarks. However, an appendix contains some examples that might be helpful for the reader.

## 2 Alternating-time Temporal Logic

ATL is not only a temporal logic, but also a game logic. It allows one to model a system interacting with an environment by representing all the system components as well as the environment as agents, evolving in a game. In this game, agents and/or coalitions of agents are given objectives to reach whatever the actions of other agents are, and the playground is a Concurrent Game Structure — in short, a CGS — which is a state space. Below, we shortly recall ATL syntax and semantics following [4] and [6].

Let $\mathcal{AP}$ be a finite set of propositions and $\Sigma$ be a finite set of symbols, the “agents”. ATL-formulas are defined by the following grammar:

$$\varphi ::= p \mid \neg \varphi \mid (\varphi_1 \land \varphi_2) \mid \langle \langle A \rangle \rangle \bigcirc \varphi \mid \langle \langle A \rangle \rangle \square \varphi \mid \langle \langle A \rangle \rangle \varphi_1 U \varphi_2$$
where \( p \in \mathcal{AP} \) and \( A \in \mathcal{P}(\Sigma) \). \( A \) is said to be a coalition of agents. The other Boolean connectives and the constant \( \top \) ("True") can be defined in the usual way.

The different parts of an ATL-formula are referred to as coalition quantifier for the expression \( \langle \langle A \rangle \rangle \) and as temporal operators for the expressions \( \bigcirc \) (next), \( \square \) (always) and \( \mathcal{U} \) (until). We can notice that in ATL syntax every temporal operator has to be preceded by a coalition quantifier. For instance \( \langle \langle A \rangle \rangle \bigcirc p \land \square q \), where \( p, q \in \mathcal{AP} \), is not an ATL-formula. This pair of coalition quantifier and temporal operator is called modal operator.

Formulas of the form \( \langle \langle A \rangle \rangle \square \phi \) or \( \neg \langle \langle A \rangle \rangle \phi_1 \mathcal{U} \phi_2 \) are called invariants and formulas of the form \( \langle \langle A \rangle \rangle \phi_1 \mathcal{U} \phi_2 \) or \( \neg \langle \langle A \rangle \rangle \square \phi \) are called eventualities.

Formulas are interpreted on particular Kripke structures called concurrent game structures. A CGS is a tuple \( \mathcal{M} = (\Sigma, S, \mathcal{AP}, L, d, \delta) \) with the following components:

1. \( \Sigma \) is a finite, non-empty set of agents, referred to by the numbers 1 through \( n = |\Sigma| \).
2. Subsets of \( \Sigma \) are called coalitions.
3. \( S \) is a set of states.
4. \( \mathcal{AP} \) is a finite set of propositions.
5. \( L \) is a labeling function. \( L \) maps each state \( s \in S \) to a set \( L(s) \subseteq \mathcal{AP} \) of propositions true at \( s \).
6. For each agent \( a \in \Sigma \) and each state \( s \in S \), \( d_a(s) \geq 1 \) is the number of moves available at state \( s \) to agent \( a \). The moves of agent \( a \) are identified with the numbers 0 through \( d_a(s) - 1 \). For every state \( s \in S \), a move vector \( \sigma \) at \( s \) is a \( n \)-tuple \( (\sigma_1, \ldots, \sigma_n) \) such that \( 0 \leq \sigma_a \leq d_a(s) - 1 \) for each agent \( a \). Given a state \( s \in S \), we denote by \( D_a(s) \) the set \( \{0, \ldots, d_a(s) - 1\} \) and by \( D(s) \) the set \( \{0, \ldots, d_1(s) - 1\} \times \ldots \times \{0, \ldots, d_n(s) - 1\} \) of move vectors at \( s \). The function \( D \) is called move function, while \( \delta \) is a transition function, mapping each state \( s \in S \) and each move vector \( \sigma = (\sigma_1, \ldots, \sigma_n) \in D(s) \) to a state \( \delta(s, \sigma_1, \ldots, \sigma_n) \in S \).

Below, we refer to an arbitrarily fixed concurrent game structure \( \mathcal{M} \).

**Definition 1.** Let \( s, s' \in S \) be two states. State \( s' \) is a successor of \( s \) if \( s' = \delta(s, \sigma) \) for some \( \sigma \in D(s) \). A computation of \( \mathcal{M} \) is an infinite sequence \( \lambda = s_0, s_1, \ldots \) of states of \( S \) such that, for all \( i \geq 0 \), the state \( s_{i+1} \) is a successor of the state \( s_i \). Elements of the domain of \( \lambda \) (i.e. 0, 1,...) are called positions. For a computation \( \lambda \) and positions \( i, j \geq 0 \), \( \lambda[i] \) and \( \lambda[j, i] \) denote the \( i \)th state of \( \lambda \) and the finite sequence \( s_j, s_{j+1}, \ldots, s_i \) of \( \lambda \), respectively.

Let \( s \in S \) and let \( A \subseteq \Sigma \) be a coalition of agents. An A-move \( \sigma_A \) at state \( s \) is a \( |\Sigma| \)-tuple such that \( \sigma_A(a) \in D_a(s) \) for every \( a \in A \) and \( \sigma_A(a') \) is any \( \sigma(a') \in SD_a'(s) \) for every \( a' \notin A \). \( D_A(s) \) denotes the set of all A-moves at state \( s \). A move vector \( \sigma \) extends an A-move — written \( \sigma_A \subseteq \sigma \) — if \( \sigma(a) = \sigma_A(a) \) for every \( a \in A \).

The outcome of \( \sigma_A \in D_A(s) \) at state \( s \), denoted by \( \text{out}(s, \sigma_A) \), is the set of all states \( s' \) for which there exists a move vector \( \sigma \in D(s) \) such that \( \sigma_A \subseteq \sigma \) and \( \delta(s, \sigma) = s' \).
A coalition of agents $A$, in order to achieve its objective, uses strategies to decide an $A$-move to play at any state of the concurrent game structure. In general, at any given state $s$, the strategical choice of $A$ may depend on some part of the history of the computation. In [4], it is proved that, for ATL, the choice between the two extreme hypotheses concerning memory of the agents — namely: they can remember any past state or else be memoryless — is irrelevant to satisfiability of a formula. In the following, we formally define and use only those strategies that are based just on knowledge of the current state. This kind of strategies are called positional strategies.

A positional strategy for a coalition $A \subseteq \Sigma$ — or $A$-strategy — is a mapping $F_A : S \mapsto \bigcup \{ D_A(s) \mid s \in S \}$ such that $F_A(s) \in D_A(s)$ for every $s \in S$.

The outcome of an $A$-strategy $F_A$ at state $s$, denoted by $\text{out}(s, F_A)$, is the set of all computations $\lambda$ such that $\lambda[0] = s$ and $\lambda[i + 1] \in \text{out}((\lambda[i], F_A(\lambda[i])))$, for all $i \geq 0$.

We can finally give the semantics of ATL-formulas. Below, we only give the clauses specific to ATL. Formally, $\mathcal{M}, s \models \varphi$ denotes that the state $s$ satisfies the formula $\varphi$ in the structure $\mathcal{M}$. For all states $s \in S$, the satisfaction relation $\models$ is inductively defined as follows:

1. $\mathcal{M}, s \models \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there exists an $A$-move $\sigma_A$ such that, for all $s' \in \text{out}(s, \sigma_A)$, $\mathcal{M}, s' \models \varphi$.
2. $\mathcal{M}, s \models \langle\langle A \rangle\rangle \square \varphi$ iff there exists an $A$-strategy $F_A$ such that, for all computations $\lambda \in \text{out}(s, F_A)$ and all positions $i \geq 0$, $\mathcal{M}, \lambda[i] \models \varphi$.
3. $\mathcal{M}, s \models \langle\langle A \rangle\rangle \varphi_1 U \varphi_2$ iff there exists an $A$-strategy $F_A$ such that, for all computations $\lambda \in \text{out}(s, F_A)$, there exists a position $i \geq 0$ where $\mathcal{M}, \lambda[i] \models \varphi_2$ and, for all positions $0 \leq j < i$, $\mathcal{M}, \lambda[j] \models \varphi_1$.

A formula $\theta$ is satisfiable when there are a concurrent game structure $\mathcal{M}$ and a state $s$ such that $\mathcal{M}, s \models \theta$. When the agents are bound to agents occurring in $\theta$, then one says that $\theta$ is tight satisfiable (see[6]). The interested reader might consult that work to see a discussion of several possible notions of ATL-satisfiability.

When $\Gamma$ is a set of ATL-formulas, $\mathcal{M}, s \models \Gamma$ iff, for all $\varphi \in \Gamma$, $\mathcal{M}, s \models \varphi$.

ATL-formulas can be decomposed into “semantically simpler” formulas. The semantically simplest ones will be called primitive formulas.

An ATL-formula $\varphi$ is primitive if it has one of the following forms: a) either $\top$, $p$, or $\neg p$ where $p \in AP$, that is a literal; b) $\langle\langle A \rangle\rangle \bigcirc \varphi'$ with $\varphi'$ a formula, that is a positive next-time formula; c) $\neg \langle\langle A \rangle\rangle \bigcirc \varphi'$ with $\varphi'$ a formula and $A \neq \Sigma$, that is a negative next-time formula. Formulas of the forms (b) and (c) are called next-time formulas.

Every non-primitive formula is classified as $\alpha$-formula or as $\beta$-formula and, as usual, $\alpha \equiv \alpha_1 \land \alpha_2$ and $\beta \equiv \beta_1 \lor \beta_2$. The components $\alpha_1$, $\alpha_2$, $\beta_1$ and $\beta_2$ depends on $\alpha$ and $\beta$ in the usual way for Boolean formulas, and the classification is extended to typical ATL-formulas similarly to how it is done for CTL formulas, as follows:
Then a set $\Delta$ of ATL-formulas is **downward saturated** if the following conditions are satisfied: 1) if $\alpha \in \Delta$, then $\alpha_1, \alpha_2 \in \Delta$, 2) if $\beta \in \Delta$, then $\beta_1 \in \Delta$ or $\beta_2 \in \Delta$.

Let $\Gamma$ and $\Delta$ be sets of ATL-formulas. We say that $\Delta$ is a **minimal downward saturated extension of $\Gamma$** if $\Gamma \subseteq \Delta$, $\Delta$ is downward saturated and there is no downward saturated set $\Delta'$ such that $\Gamma \subseteq \Delta' \subset \Delta$.

As usual for modal logics, a notion of closure of a given formula is also defined:

**Definition 2.** Let $\theta$ be an ATL-formula. The closure of $\theta$, denoted by $\text{cl}(\theta)$, is the least set of formulas such that

1. $\theta \in \text{cl}(\theta)$
2. $\text{cl}(\theta)$ is closed under subformulas
3. if $\langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2 \in \text{cl}(\theta)$, then $\varphi_1 \land \langle\langle A\rangle\rangle \circ \langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2 \in \text{cl}(\theta)$
4. if $\neg\langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2 \in \text{cl}(\theta)$, then $\neg\varphi_2 \land \neg\varphi_1, \neg\varphi_2 \land \neg\langle\langle A\rangle\rangle \circ \langle\langle A\rangle\rangle \varphi_1 \mathcal{U} \varphi_2 \in \text{cl}(\theta)$
5. if $\langle\langle A\rangle\rangle \Box \varphi \in \text{cl}(\theta)$, then $\varphi \land \langle\langle A\rangle\rangle \circ \langle\langle A\rangle\rangle \Box \varphi \in \text{cl}(\theta)$.

### 3 Methods for deciding ATL satisfiability

In this section we recall and compare two existing methods to test ATL satisfiability. We invite the readers to consult the cited original works for a better understanding.

#### 3.1 Decision by means of $q$-ary alternating tree automata

[7] proves the decidability of the problem of satisfiability for an ATL formula by reducing it to the non-emptiness problem for alternating automata on infinite trees. In order to describe such a reduction, we previously recall some standard definitions.

A **tree** is a set $T \in \mathbb{N}^*$ such that if $t.c \in T$ where $t \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $t \in T$, and for all $0 \leq c' \leq c$, $t.c' \in T$. The elements of $T$ are called **nodes** and the empty word $\varepsilon$ is the **root** of $T$. For every $t \in T$, the nodes $t.c$ where $c \in \mathbb{N}$ are the **successors** of $t$. The **degree** of $t$, $d(t)$, is the number of successors of $t$. The **successor function** $\text{succ} : T \mapsto 2^T$ maps each node $t \in T$ to the set $\text{succ}(t)$.
A node is a leaf if it has no successors, otherwise it is an interior node. Given an alphabet $\Theta$, a $\Theta$-labeled tree — or computation tree — is a pair $(T, V)$ where $T$ is a tree and $V : T \mapsto \Theta$ maps each node of $T$ to a symbol in $\Theta$.

Transitions of alternating tree automata use positive Boolean formulas, that is formulas built over a set $X$ of Boolean variables by using only connectors $\land$ and $\lor$, and possibly True and False. Let $X$ be a given set of propositional variables. The set of positive Boolean formulas over $X$ is noted $\mathcal{B}^+(X)$. Let $Y$ be a subset of $X$ and let $\vartheta \in \mathcal{B}^+(X)$ be a positive Boolean formula. One says that $Y$ satisfies $\vartheta$ if and only if assigning True to elements in $Y$ and assigning False to elements in $X \setminus Y$ makes $\vartheta$ true.

The next definitions are based on [9] and [7]. An alternating tree automaton on infinite trees is a tuple $A = (\Theta, D, Q, I, \rho, F)$ where $\Theta$ is a finite alphabet, $D \subset \mathbb{N}$ is a finite set of arities, $Q$ is a finite set of states, $I \subseteq Q$ is a set of initial states, $\rho : Q \times \Theta \times D \mapsto \mathcal{B}^+ \{(0, \ldots, d - 1) \times Q\}$, with $d \in D$, is a partial transition function, and $F \subseteq Q$ is the set of final (accepting) states.

**Example 1 (Alternating tree automaton).** The automaton $A = (\Theta, D, Q, I, \rho, F)$ is defined as follows:

1. $\Theta = \{a, b\}$
2. $D = \{2\}$
3. $Q = \{q_1, q_2, q_3\}$
4. $I = \{q_1\}$
5. $F = \{q_1\}$
6. The transition function $\rho$
   - (a) $\rho(q_1, a) = (0, q_1) \lor (1, q_2)$
   - (b) $\rho(q_1, b) = (0, q_2)$
   - (c) $\rho(q_2, a) = \text{true}$
   - (d) $\rho(q_2, b) = (0, q_1) \land (1, q_3)$
   - (e) $\rho(q_3, a) = ((1, q_1) \lor (1, q_2)) \land (0, q_3)$
   - (f) $\rho(q_3, b) = \text{true}$

An alternating tree automaton runs over an infinite $\Theta$-labeled tree $(T, V)$ and produces a tree $(T_r, r)$. Each node of $(T_r, r)$ is labeled by an element of $\mathbb{N}^* \times Q$, that is a node $t$ of $(T, V)$ and a state $q$ of the alternating tree automaton. A node in $T_r$ labeled by $(t, q)$ describes a copy of the automaton that reads the node $t$ of $T$ in state $q$. Formally, a run of an automaton $A = (\Theta, D, Q, I, \rho, F)$ is defined as follows:

A run $(T_r, r)$ of an automaton $A$ over a $\Theta$-labeled tree $(T, V)$ is a $\Theta_r$-labeled tree where $\Theta_r = \mathbb{N}^* \times Q$, and $(T_r, r)$ satisfies the following:

1. $r(\varepsilon) = (\varepsilon, q_0)$ where $q_0 \in I$.
2. Let $y \in T_r$ with $r(y) = (t, q)$ and $\rho(s, V(t), d(t)) = \vartheta$. Then there is a (possibly empty) set $\Omega = \{(c_0, q_0), (c_1, q_1), \ldots, (c_n, q_n)\} \subseteq \{0, \ldots, d(t) - 1\} \times Q$, such that the following holds:
   - (a) $\Omega$ satisfies $\vartheta$.
   - (b) For all $0 \leq i \leq n$, $y.i \in T_r$ and $r(y.i) = (t.c_i, q_i)$.

A run $(T_r, r)$ of $A$ over a tree is accepting if all its finite paths terminate by true and all its infinite paths $\lambda$ satisfy $\operatorname{inf}(\lambda) \cap F \neq \emptyset$, where $\operatorname{inf}(\lambda)$ is the set of
states appearing infinitely many times in $\lambda$. An automaton accepts a tree if and only if the tree is accepted by at least one run of the automaton.

In the particular case where $D$, the set of arities of an automaton $A$, is a singleton $\{K\}$ for some given natural number $K$, the automaton will read only input trees $(T, V)$ having a constant branching factor $K$, that is, each node will have exactly $K$ successors. We will call such trees $K$-ary trees and we will say that $A$ is a $K$-ary alternating automaton.

Let us observe that an interpretation $M$ for an ATL formula $\theta$, based on a CGS, is either an infinite graph (when the set $S$ of states is infinite) or else a finite graph with cycles. In both cases unwinding $M$ produces an infinite labeled tree where nodes are labeled by proposition sets and edges are labeled by move vectors. Alternating tree automata, however, can read only trees having unlabeled edges. In the case where $M$ is such that each agent has exactly the same number of moves available at each state, unwinding $M$ produces a $K$-ary tree, thus the information provided by edge labels can be easily encoded at the level of tree nodes.

The procedure associating an alternating automaton $A_\theta$ to a formula $\theta$ defined in [7], and recalled below, follows this principle. The automaton states are elements of the closure of $\theta$, $\theta$ itself is the initial state and the final states are the invariant formulas. In the following definition, due to [7], as the set of move vectors is the same regardless of the node, out($\sigma$) stands for out($\sigma, t$).

**Definition 3 ([7]).** Given a formula $\theta$, the elements of the automaton $A_\theta = \langle Q, D, Q, I, \rho, F \rangle$ are the following:

1. $Q$ is the set of atomic propositions of $\theta$ and $Q = AP$.
2. $D$ contains only one element $k^n$ where $k = |\Psi_\theta| + 1$ with $\Psi_\theta = \{\varphi \in \mathcal{E}(\theta) | \varphi = \langle A \rangle \circ \varphi' \text{ or } \varphi = \neg\langle A \rangle \circ \varphi', \text{ and } n \text{ the number of agents in } \theta\}$. $Q = \mathcal{E}(\theta)$. Each state of $A_\theta$ are named by an element of the closure of $\theta$.
3. $I = \theta$.
4. $F$ contains states of $Q$ named by an invariant formula.
5. The transition function $\rho$ is defined for all $\pi \in 2^V$ as follows:
   - (a) $\rho(p, \pi) = true$ if $p \in \pi$
   - (b) $\rho(p, \pi) = false$ if $p \notin \pi$
   - (c) $\rho(\neg p, \pi) = false$ if $p \in \pi$
   - (d) $\rho(\neg p, \pi) = true$ if $p \notin \pi$
   - (e) $\rho(\langle A \rangle \varphi_1 \varphi_2, \pi) = \rho(\varphi_1, \pi) \land \rho(\varphi_2, \pi)$
   - (f) $\rho(\langle A \rangle \varphi_1 \lor \varphi_2, \pi) = \rho(\varphi_1, \pi) \lor \rho(\varphi_2, \pi)$
   - (g) $\rho(\langle A \rangle \circ \varphi, \pi) = V_{\sigma \in \Delta A} \left( \Lambda_{c \in \text{out}(\sigma)} (c, \varphi) \right)$
   - (h) $\rho(\neg\langle A \rangle \circ \varphi, \pi) = \Lambda_{c \in \Delta A} \left( V_{\sigma \in \text{out}(\sigma)} (c, \neg \varphi) \right)$
   - (i) $\rho(\langle A \rangle \Box \varphi, \pi) = \rho(\varphi, \pi) \land \rho(\langle A \rangle \circ \langle A \rangle \Box \varphi, \pi)$
   - (j) $\rho(\neg\langle A \rangle \Box \varphi, \pi) = \rho(\neg \varphi, \pi) \lor \rho(\langle A \rangle \circ \langle A \rangle \Box \varphi, \pi)$
   - (k) $\rho(\langle A \rangle \varphi_1 U \varphi_2, \pi) = \rho(\varphi_2, \pi) \lor \rho(\varphi_2, \pi) \land \rho(\langle A \rangle \circ \langle A \rangle \varphi_1 U \varphi_2, \pi))$
   - (l) $\rho(\neg\langle A \rangle \varphi_1 U \varphi_2, \pi) = \rho(\neg \varphi_2, \pi) \land (\rho(\neg \varphi_1, \pi) \lor \rho(\neg\langle A \rangle \circ \langle A \rangle \varphi_1 U \varphi_2, \pi))$
The inputs of $\mathcal{A}_\theta$ are $K$-ary $\Theta$-labeled trees where $\Theta$ is the power-set of $\mathcal{AP}$ (the propositions in $\theta$). The value of the constant branching degree $K$ is $k^n$ where $n = |\Sigma|$ and $k$ is the cardinality of the set $\dim\Psi \circ \dim$ of all next-time formulas in the closure of $\theta$ plus 1. It is worthwhile observing that when tight satisfiability is the notion of interest, as in our case, all the $n$ agents are named in $\theta$, thus $K$ actually depends only on $\theta$. Any of these trees can be seen as describing a concurrent game structure, possibly with an infinite number of states, where actions of agents consist in choosing a next-formula and each successor node represents the result of a move vector. Any given successor of a node $t$, say $t.c$, is such that the number $c$, when taken to be written in a $k$-based number system, encodes a move vector $\sigma = \langle j_1, \ldots, j_n \rangle$ where $1 \leq i \leq n, 0 \leq j_i \leq k - 1$. Indeed, [7] defines a natural bijection $\tau$ between the set of all possible vectors of actions of the $n$ agents and the set $\{0, \ldots, k^n - 1\}$ of numbers written with base $k$.

**Definition 4.** The bijective encoding $\tau : \{0, \ldots, k - 1\}^n \mapsto \{0, \ldots, k^n - 1\}$ is defined by:

$$
\tau(\sigma) = j_1k^{n-1} + j_2k^{n-2} + \cdots + j_nk^0
$$

The function $\tau$ is a bijection between the set of all possible vectors of actions of the $n$ agents — each action being the choice of one of the $k$ next-time formulas — and the set $\{0, \ldots, k^n - 1\}$ of numbers written with base $k$. Hence, $\tau(\sigma)$ uniquely encodes a move vector $\sigma$, while its inverse allows one to recover the vector out of a number $c : \tau^{-1}(c) = \sigma$. It is worthwhile noticing that the use of such an encoding of vectors exploits the assumption that every agent has the same choices. Thus, when an input of the automaton is seen as being a CGS itself, the set $D_A(t)$ of $A$-moves for a coalition $A \subseteq \Sigma$ is constant. $\Delta_A$ notes the set of $A$-moves for a coalition $A \subseteq \Sigma$ in a labeled tree with fixed branching degree and $\text{out}(\sigma_A)$ is the output set of move vectors $\sigma$ where $\sigma_A \in \Delta_A$ and $\sigma_A \subseteq \sigma$.

A CGS encoded by a $K$-ary tree as described above is a particular kind of structure. [7] proves that satisfiability of a formula $\theta$ can be reduced to non-emptiness of the tree language recognized by $\mathcal{A}_\theta$. Actually also something more is proved: if a tree $T$ is accepted then a finite model of $\theta$ can be defined out of $T$, thereby obtaining a finite model property.

### 3.2 Goranko and Shkatov’s Tableau Method

In [6] a tableau method to test tight satisfiability for ATL formulas is defined. The new idea is to build candidate models for a formula in such a way that, when constructing the successors of a state, one can avoid to build $a$ priori useless nodes, thereby reducing the search space for models. A tableau, here, is a graph where nodes are labeled by sets of formulas and edges are given

---

1 In [7], Remark 26 says: The CGS so constructed is a “Moore synchronous” CGS in terms of [3] – the state space is composed of the product of local components, one for each player. The evolution of the system proceeds in “lock-step” with every player determining its next local state.
by binary relations \( \sigma \rightarrow \), with \( \sigma \) a move vector. However, a first phase of the tableau procedure, called \textit{construction phase}, generates first a pretableau, where an additional kind of link between nodes, noted \( \sigma \Rightarrow \), is also present. A set of formulas labeling a pretableau node can be of two kinds: a \textit{state} or a \textit{prestate} (the embryo of states), and a prestate is connected to states by \( \Rightarrow \) edges. At the end of the tableau procedure only states will remain, and the pruning of the graph (eliminating also \( \Rightarrow \) edges) is called \textit{elimination phase}. This phase acts on a pretableau by eliminating all prestates and, then, iteratively, all inconsistent states, all states containing eventualities that cannot be realized and all states which have lost all successors linked with the same \( \sigma \rightarrow \).

The construction phase generating a pretableau for an ATL formula \( \theta \) starts by creating, as initial node, the prestate \( \Gamma = \{ \theta \} \), then applies as far as possible two rules: a static one, \textbf{SR}, that takes a prestate as input and creates a state out of it, by saturating it, and a dynamical one, the rule \textbf{Next}, that takes a state as input and creates prestates in sufficient number to enforce the truth of all “Next-time formulas”.

In the sequel, we detail the construction rules.

\textbf{Rule SR.} Given a prestate \( \Gamma \), the steps of the rule \textbf{SR} are:

1. add to the pretableau all the minimal downward saturated extensions \( \Delta \) of \( \Gamma \) as states
2. For each of the so obtained states \( \Delta \):
   (a) set \( \Gamma \Rightarrow \Delta \) and if \( \Delta \) does not contain any formulas of the form \( \langle\langle A \rangle \rangle \bigodot \varphi \) or \( \neg \langle\langle A \rangle \rangle \bigodot \varphi \), add the formula \( \langle\langle \Sigma \theta \rangle \rangle \bigodot \top \).
   (b) If, however \( \Delta \) coincides with a state \( \Delta' \) of the pretableau, do not create another copy of \( \Delta' \) but set \( \Gamma \Rightarrow \Delta' \) (loop checking step).

\textbf{Rule Next.} Given a state \( \Delta \) not containing a pair of formulas \( \gamma \), \( \neg \gamma \), the steps of rule \textbf{Next} are:

1. Order linearly all positive and negative next-time formulas of \( \Delta \) in such a way that all positive next-time formulas precede all the negative ones, thereby obtaining a list:
   \[ L = \{ \langle\langle A_0 \rangle \rangle \bigodot \varphi_0, \ldots, \langle\langle A_m-1 \rangle \rangle \bigodot \varphi_{m-1}, \neg\langle\langle A'_0 \rangle \rangle \bigodot \psi_0, \ldots, \neg\langle\langle A'_{l-1} \rangle \rangle \bigodot \psi_{l-1} \}. \]
   Let \( r_\Delta \) be the sum \( m + l \) and \( D(\Delta) \) be the set \( \{0, \ldots, r_\Delta - 1\}^{\Sigma_\theta} \). Also, for every \( \sigma \in D(\Delta) \), set \( N(\sigma) \) to be the set \( \{ i \mid \sigma_i \geq m \} \), where \( \sigma - i \) stands for the \( i \)th component of the tuple \( \sigma \) and set \( \neg \sigma = \{ i \mid \sigma_i < m \} \).
2. Consider the elements of \( D(\Delta) \) in the lexicographic order and for each \( \sigma \in D(\Delta) \) do the following:
   (a) Create a prestate
   \[ \Gamma_\sigma = \{ \varphi_p \mid \langle\langle A_p \rangle \rangle \bigodot \varphi_p \in \Delta \text{ and } \sigma_a = p \text{ for all } a \in A_p \} \]
   \[ \cup \{ \neg \psi_q \mid \neg\langle\langle A'_q \rangle \rangle \bigodot \psi_q, \neg \sigma = q, \text{ and } (\Sigma_\theta - A'_q) \subseteq N(\sigma) \} \]
or put $\Gamma_{\sigma} = \{ \top \}$ if the sets on both sides of the union sign above is empty.

(b) Connect $\Delta$ to $\Gamma_{\sigma}$ with $\sigma \rightarrow$. If, however, $\Gamma_{\sigma} = \Gamma$ for some prestate of the pretableau, do not create a copy of $\Gamma$ and only connect $\Delta$ to $\Gamma$ with $\sigma \rightarrow$.

Intuitively, the rule Next, when applied to a state $\Delta$, treats its primitive next-time formulas, and produces all necessary prestates which will evolve (by the means of SR) in the successors of $\Delta$ enforcing the truth of the next-time formulas. To help the intuitive understanding of this rule, is worthwhile noticing that $r_\Delta$ is the total number of next-time formulas in $\Delta$, positive as well as negative. Next-time formulas are ordered in the list $L$ and enumerated from $0$ to $r_\Delta - 1$: let us call this enumeration “global enumeration”. Thus, the set $D(\Delta)$ corresponds to the set of all move vectors where actions of agents consist in choosing a next-time formula by providing its number in the global enumeration. Therefore, given any $\sigma \in D(\Delta)$, $N(\sigma)$ is the set of agents having chosen a negative next-time formula. The number $\text{neg}(\sigma)$, however, associates to any given vector $\sigma$ the number of a negative formula according to the “auxiliary enumeration” numbering negative formulas from $0$ to $l - 1$. The underlying idea is that Next connects $\Delta$ via any given vector $\sigma$ to a successor prestate, containing $\varphi_p$, whenever all the members of the coalition $A_p$ involved in the positive next-time formula $\langle \langle A_p \rangle \rangle \chi \varphi_p$ “vote”, via $\sigma$, to make it true. The treatment of negative formulas is subtler and rather technical. We refer the readers to [6] for explanations of this point. That work shows that only satisfiable prestates can be constructed from a satisfiable state. It is worth noticing that such an approach does not construct all the possible successors of a tableau state, but just necessary ones: it cuts away a priori some choices that certainly would not work, i.e. cuts away some “trivially” inconsistent choices.

The construction phase ends when no new states can be added by the above rules; the pretableau is then said to be complete. A pretableau for $\theta$ is noted $P^\theta$. A complete pretableau $P^\theta_1$ for the formula $\theta_1 = \neg \langle \langle 1 \rangle \rangle \boxdot p \land \langle \langle 1, 2 \rangle \rangle \land p \land \neg \langle \langle 2 \rangle \rangle \land \neg p$ is available in the appendix, figure 3.

Given a state $\Delta$, prestates($\Delta$) is the set of all prestates $\Gamma$ such that $\Delta \sigma \rightarrow \Gamma$ for some $\sigma \in D(\Delta)$. Given a prestate $\Gamma$, the set of all states $\Delta$ such that $\Gamma \Rightarrow \Delta$ is noted states($\Gamma$).

The elimination phase prunes $P^\theta$ by first removing any prestate $\Gamma$ (and the $\Rightarrow$ edges coming out of it) and replacing it by the states generated by $\Gamma$; this is achieved by the rule (PR). Then, “bad” states are eliminated. A state $\Delta$ needs to be cut off because one of three reasons: rule E1 eliminates states containing a patent inconsistency, rule E2 eliminates states which have lost all their successors linked with the same $\rightarrow$, because of previous eliminations, and rule E3 eliminates states containing an eventuality that is not realized in the tableau. The elimination rules (E1), (E2) and (E3) are iteratively applied as much as possible. The resultant tableau is called the final tableau for $\theta$ and is denoted by $T^\theta$. If $\theta \in \Delta$ for some $\Delta \in S^\theta$, then the final tableau $T^\theta$ is said open, otherwise closed.
Below, we detail the elimination rules.

The first step of the elimination phase is the application of the rule **Rule (PR):**
For every prestate $\Gamma$ in $\mathcal{P}^\theta$, do the following:

1. For all states $\Delta \in \mathcal{P}^\theta$ with $\Delta \xrightarrow{\sigma} \Gamma$ and all $\Delta' \in \text{states}(\Gamma)$, put $\Delta \xrightarrow{\sigma} \Delta'$
2. Remove $\Gamma$ from $\mathcal{P}^\theta$.

The rule **PR** produces a graph called *initial tableau* and denoted $\mathcal{T}_0^\theta$.

The next step is to eliminate from the initial tableau $\mathcal{T}_0^\theta$ all states that cannot be satisfied in any model of $\theta$. At each elimination with rules (E1) – (E3), an intermediate tableau $\mathcal{T}_{n+1}^\theta$ is produced from $\mathcal{T}_n^\theta$. $S_n^\theta$ is the set of states in intermediate tableau $\mathcal{T}_n^\theta$.

**Rule (E1).** If $\{\varphi, \neg \varphi\} \subseteq \Delta \in S_n^\theta$, then obtain $\mathcal{T}_{n+1}^\theta$ by eliminating $\Delta_1$ from $\mathcal{T}_n^\theta$.

**Rule (E2).** If, for some $\sigma \in D(\Delta)$, all states $\Delta'$ with $\Delta \xrightarrow{\sigma} \Delta'$ have been eliminated already, then obtain $\mathcal{T}_{n+1}^\theta$ by eliminating $\Delta$ from $\mathcal{T}_n^\theta$.

To formulate the rule **E3**, it is necessary to previously technically define what means to realize an eventuality in a tableau.

**Definition 5.**
1. $\text{vect}(\Delta, (\langle A \rangle) \circ \varphi_{a}) = \{\sigma \in D(\Delta) \mid \sigma_a = p \text{ for every } a \in A\}$
2. $\text{vect}(\Delta, \neg (\langle A' \rangle) \circ \varphi_{q}) = \{\sigma \in D(\Delta) \mid \text{neg}(\sigma) = q \text{ and } \Sigma^\theta \setminus A' \subseteq N(\sigma)\}$

**Definition 6.** Realization of eventuality $(\langle A \rangle) \varphi_1 \cup \varphi_2$.
1. If $\{\varphi_2, (\langle A \rangle) \varphi_1 \cup \varphi_2\} \subseteq \Delta \in S_n^\theta$, then $(\langle A \rangle) \varphi_1 \cup \varphi_2$ is realized at $\Delta$ in $\mathcal{T}_n^\theta$.
2. If $\{\varphi_1, (\langle A \rangle) \circ (\langle A \rangle) \varphi_1 \cup \varphi_2, (\langle A \rangle) \varphi_1 \cup \varphi_2\} \subseteq \Delta$ and for every $\sigma \in \text{vect}(\Delta)$, $(\langle A \rangle) \circ (\langle A \rangle) \varphi_1 \cup \varphi_2$, there exists $\Delta' \in S_n^\theta$ such that
   a. $\Delta \xrightarrow{\sigma} \Delta'$ and
   b. $(\langle A \rangle) \varphi_1 \cup \varphi_2$ is realized at $\Delta'$ in $\mathcal{T}_n^\theta$,
then $(\langle A \rangle) \varphi_1 \cup \varphi_2$ is realized at $\Delta$ in $\mathcal{T}_n^\theta$.

**Definition 7.** Realization of eventuality $\neg (\langle A \rangle) \square \varphi$.
1. If $\{\neg \varphi, (\langle A \rangle) \square \varphi\} \subseteq \Delta \in S_n^\theta$, then $\neg (\langle A \rangle) \square \varphi$ is realized at $\Delta$ in $\mathcal{T}_n^\theta$.
2. If $\{\neg (\langle A \rangle) \circ (\langle A \rangle) \varphi, (\langle A \rangle) \square \varphi\} \subseteq \Delta$ and for every $\sigma \in \text{vect}(\Delta, \neg (\langle A \rangle) \circ (\langle A \rangle) \varphi)$, there exists $\Delta' \in S_n^\theta$ such that
   a. $\Delta \xrightarrow{\sigma} \Delta'$ and
   b. $\neg (\langle A \rangle) \square \varphi$ is realized at $\Delta'$ in $\mathcal{T}_n^\theta$,
then $\neg (\langle A \rangle) \square \varphi$ is realized at $\Delta$ in $\mathcal{T}_n^\theta$.

**Rule (E3).** If $\Delta \in S_n^\theta$ contains an eventuality that is not realized at $\Delta \in \mathcal{T}_n^\theta$, then obtain $\mathcal{T}_{n+1}^\theta$ by removing $\Delta$ from $\mathcal{T}_n^\theta$. 

Example 2. Let $\theta$ be $\neg\langle\langle 1 \rangle\rangle \Box p \land \langle\langle 1, 2 \rangle\rangle \circ p \land \neg\langle\langle 2 \rangle\rangle \circ \neg p$.

As no states have to be removed with rules (E1), (E2) or (E3) then $T^{\theta_1} = T^{\theta_1}_0$. Moreover, $\theta_1 \in \Delta_1$ and $\Delta_2$ so $T^{\theta_1}$ is open. The final tableau, in this specific case, is the same as what we call, in the next section semi-final tableau for $\theta_1$ and it is given in the Appendix, figure 5.

As proved in [6], this tableau system terminates and is sound and complete with respect to tight ATL-satisfiability with positional strategies, that is, any complete final tableau $T^{\theta}$ is open if and only if $\theta$ is tight-satisfiable. The system can be easily modified so as to deal with other notions of satisfiability (see [6]).

4 Tableau construction as automata construction

In this section we show how the procedure generating tableaux testing ATL satisfiability as described in [6] provides also a way to build an alternating automaton for ATL formulas, thereby establishing a connection between two tools for analyzing ATL formulas – alternating tree automata and tableaux – that have been independently proposed.

Let us call semi-final tableau for a formula $\theta$, and note $ST^{\theta}$, a tableau obtained from a complete pretableau for $\theta$ by pruning it, in the elimination phase, without using the rule (E3). A naive way of connecting the tableau construction to the automaton construction might consist, essentially, in taking the states of $ST^{\theta}$ as the states of the automaton. Below, we show to construct such an automaton, that we note $A'_{\theta}$.

Given a state $q$ of $ST^{\theta}$ and a vector $\sigma \in D(q)$, let us note succ$^\sigma(q)$ the set of all states $q'$ such that $q \xrightarrow{\sigma} q'$. The automaton $A'_{\theta}$ is defined by the tuple $\langle \Theta, D, Q, I, \rho, F \rangle$ where:

1. $\Theta = \text{AP}$, the set of atomic propositions in $\theta$.
2. $D$ contains only one element $k^\theta$ where $k = |\Psi_\circ| + 1$ with $\Psi_\circ = \{ \varphi \in \text{cl}(\theta) | \varphi = \langle\langle A \rangle\rangle \circ \varphi' \text{ or } \varphi = \neg\langle\langle A \rangle\rangle \circ \varphi' \}$, and $n$ the number of agents in $\theta$.
3. $Q = S^{\theta}$ is the set of states of $ST^{\theta}$, excluding the state $\{\top, \langle\langle \Sigma \rangle\rangle \circ \top\}$.
4. $I \subseteq Q$ is the set of states containing the ATL-formula $\theta$.
5. $F \subseteq Q$ is the set of final states; these are the states in $Q$ containing next-time formulas only of the form $\langle\langle A \rangle\rangle \circ \langle\langle A \rangle\rangle \Box \varphi$ or $\neg\langle\langle A \rangle\rangle \circ \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$.
6. The transition function $\rho$ is defined for all $\pi \in 2^{\text{AP}}$ by

$$\rho(q, \pi) = \bigwedge_{\varphi \in \text{PRIM}(q)} \mu(\varphi, \pi)$$

where $\text{PRIM}(q)$ is the set of primitive formulas in $q$, and:

(a) $\mu(p, \pi)$ = true if $p \in \pi$
(b) $\mu(p, \pi)$ = false if $p \notin \pi$
(c) $\mu(\neg p, \pi)$ = true if $p \notin \pi$
(d) $\mu(\neg p, \pi)$ = false if $p \in \pi$
(e) $\mu(\top, \pi)$ = true
\( \mu((\Sigma)) \bigcirc T, \pi) = true \)

\( \mu((A)) \bigcirc \phi', \pi) = \)

\( \forall \sigma \in \Delta_A \left( \bigwedge_{c \in \text{out}(\sigma)} \left( \forall \sigma^* \in \text{vect}(q,((A) \bigcirc \phi')) \left( \forall q' \in \text{succ}_{\sigma^*}(q)(c, q') \right) \right) \right) \)

\( \mu((\neg(A)) \bigcirc \phi', \pi) = \)

\( \forall \sigma \in \Delta_A \left( \bigwedge_{c \in \text{out}(\sigma)} \left( \forall \sigma^* \in \text{vect}(q,(\neg(A) \bigcirc \phi')) \left( \forall q' \in \text{succ}_{\sigma^*}(q)(c, q') \right) \right) \right) \)

where \( A \neq \Sigma \)

The idea behinds this automaton is to read a tree \( \langle T, V \rangle \), coding a candidate model of the formula \( \theta \) and by following the description of models — if any — in the tableau \( T^\theta \), since the tableau \( T^\theta \) describes all models satisfying \( \theta \). Let us observe that we do not construct the automaton states from \( T^\theta \), but from \( ST^\theta \). The control of eventuality realization, however, is dealt by the final states. It is important to observe that an accepting run of this automaton is such that each of its branches, say \( \lambda \), is either finite (because no invariant has been met) and ends with \( true \) or else is infinite and meets infinitely often a final state, that is a state whose only next-time formulas are of the form \( (A) \bigcirc (A) \Box \phi \) or \( (\neg(A)) \bigcirc (A) \Box \phi_1 U \phi_2 \). Whenever \( \lambda \) meets \( true \) (and stops) or else meets a final state, this means that no eventuality \( E \) so far encountered is any longer pending, that is, \( E \) has been realized.

**Proposition 1 (Soundness and Completeness).** If there exists an accepting run \( \langle T_r, r \rangle \) of the automaton \( A'_\theta \) over a \( k^n \)-branching labeled tree \( \langle T, V \rangle \), then \( \langle T, V, \varepsilon \rangle \vDash \theta \). Conversely, If there exists a \( k^n \)-labeled tree \( \langle T, V \rangle \) such that \( \langle T, V, \varepsilon \rangle \vDash \theta \), then \( A_\theta \) accepts \( \langle T, V \rangle \) with an accepting run \( \langle T_r, r \rangle \).

The proof of this property is given in the next section.

However, exactly as in the case of the automaton \( A_\theta \) defined in [7], the language recognized by \( A'_\theta \) is the class of \( K \)-ary trees seen as CGS’s where each player has exactly the same number \( K \) of moves at each state – \( K \) being the cardinality of the set of all the next-time formulas in the closure of \( \theta \) plus 1 –, and each move \( i \) consists in choosing a given next-time formula (according to some enumeration). Such an automaton certainly can decide (tight) satisfiability of \( \theta \). However, the fact that it can recognize only very special, and somewhat unnatural, models of \( \theta \) might result inconvenient when CGS’s are to be used to specify the behavior of concrete systems.

Thus, what seems more interesting is to exploit the construction of \( ST^\theta \) to build an automaton where the recognized class of models (modulo an appropriate encoding) is exactly the class of all models of \( \theta \). Such an automaton would be exploitable both to decide satisfiability and to perform model checking. The induced relation between tableaux and automata would be quite analogous to what happens in the case of LTL, where the algorithm producing a Büchi automaton for a formula \( \theta \) accepting exactly \( \theta \)'s models actually boils down to construct a Wolper tableau [10] for \( \theta \), where the pruning phase is essentially reduced to eliminate explicitly contradictory nodes. The rest of this section is devoted to illustrate this approach, by defining what we call (for reasons that will be clear in the sequel) a joker automaton.
To start with, we need to precise what is a possible input of a joker automaton. Essentially, it is the tree obtained unwinding any given CGS, without any restriction on possible moves. Thus, in the general case, we obtain a tree with a non-constant branching factor where edges are labeled by vectors. However, such a tree needs to be readable by an alternating tree automaton, that, in principle, cannot deal with labeled edges, as already observed.

The solution that we propose for this technical difficulty is to artificially complete the tree obtained by unwinding a given concurrent game structure with “fake nodes” in order to give the same number of choices to every agent at a given node, while, however, leaving open the possibility for the nodes of the associated tree to have varying degrees. In this way, it will still be possible to retrieve edge label information (i.e. vectors) by the position of a node in the tree.

Yet, just adding fake nodes and labeling all of them with a special propositional symbol, say $\text{NEN}$ (Non-Existing Node) is not yet adequate, because perturbations on the model might be generated: fake nodes might represent choices that a coalition does not really have in the considered concurrent game structure. Example 1 given in the sequel will illustrate this point. Therefore, in order to prevent the automaton accepting runs with impossible moves, we chose to add a mark to each coalition present in $\theta$ on each newly added node so as to indicate whether the coalition’s choice is possible in $M$ — mark $Y$ (Yes) — or not — mark $N$ (No). We call these artificially introduced nodes joker nodes.

A mark is associated with any coalition $A$ given in $\theta$ because, at a node $y$ with $r(y) = (t,q)$ and for each next-time $\varphi$ of the form $\varphi = (\langle A \rangle)\bigcirc \varphi'$ or $\neg (\langle A \rangle)\bigcirc \varphi'$, the automaton tries all the different possibilities for coalition $A$ to satisfy $\varphi$. Each possibility results in a set of different successor nodes $y'$ with $r(y') = (c',q')$ and certain $t.c'$ may be joker nodes. When reading a joker node with state $q'$, the automaton should know which coalition is implied in the transition between $q$ and $q'$, but this information is lost by the rule $\text{Next}$. So we need also to slightly modify the expansion rule $\text{Next}$ as well as prestates and states of the tableau procedure in such a way that, for each added prestate, the coalitions implied in the selected next-time formulas are indicated. This new tableau procedure — that, obviously, keeps sound and complete with respect to unsatisfiability of formula — makes it possible to define our automaton.

Below, we previously define Labeled Joker Trees, which are the trees read by the new automaton and we explain how to construct them from concurrent game structures. Then we indicate how to modify the tableau procedure and, finally, we describe the joker automaton.

In order to get the notion of $\Theta$-labeled joker tree, the definition of $\Theta$-labeled tree given in section 3 is modified as follows:

**Definition 8.** Given the alphabet $\Theta$ and a set of states $Q$, a $\Theta$-labeled joker tree is a triplet $\langle T, V, Q \rangle$ where $T$ is a tree, $V : T \mapsto \Theta \cup \{Y, N\}^*$ maps each node of $T$ to a letter in $\Theta$ or to a joker vector composed of symbols $Y$ and $N$, and $Q : T \mapsto Q \cup \{\emptyset\}$ maps each node of $T$ to a state in $Q$ when the node is mapped to a symbol in $\Theta$, or to $\emptyset$ otherwise.
A Θ-labeled joker tree can be used to unwind a concurrent game structure \( \mathcal{M} \). In that case, \( \Theta = 2^{AP} \) for some set \( AP \) of atomic propositions and \( Q \) is the set of states of \( \mathcal{M} \). A labeled joker tree based on \( \mathcal{M} \) is denoted by \( \langle T_M, V, Q \rangle \). The branching degree of a node \( t \) of \( T_M \) is \( d(t) = m_t^n \) where \( n = |\Sigma| \) and \( m_t = \max_{a \in \Sigma} d_a(Q(t)) \). Let \( \tau' \) be the bijection encoding defined by

\[
\tau'(\sigma, m) = j_1 m^{n-1} + j_2 m^{n-2} + \cdots + j_n m^0
\]

where \( \sigma \) is a move vector and \( m \) a natural number. During the labeled joker tree’s construction, the order of all successor nodes \( t.c \) of a node \( t \) is such that from \( c \) it is possible to obtain by means of the inverse of \( \tau' \) the successor vector associated with the node. Also, at each node \( t \) of the labeled joker tree, we can get the number \( m_t \) from the \( d(t) \) by using \( m_t = e^{\left(\log(d(t))\right)} \).

The next definition will allow the automaton to manipulate a labeled joker tree \( \langle T_M, V, Q \rangle \) coding an interpretation \( \mathcal{M} \) of a formula \( \theta \). It is worthwhile noticing that such a tree can be seen as being itself a CGS.

**Definition 9.** Let \( t \in T_M \) and let \( A \subseteq \Sigma \) be a coalition of agents, where \( |\Sigma| = n \). An \( A \)-vector \( \sigma_A \) at node \( t \) is a \( n \)-tuple \( \sigma_A \) such that \( 0 \leq \sigma_A(a) \leq m_t - 1 \) for every \( a \in A \) and \( \sigma_A(a') \) is an arbitrary fixed move of \( a' \) for every \( a' \not\in A \). \( D_A(t) \) is the set of all \( A \)-vectors \( \sigma_A \) at node \( t \). We note \( \Delta_{A,m} \) the set of \( A \)-vector \( \sigma_A \) for every node whose branching degree is \( m^n \). Let \( \sigma_A \in \Delta_{A,m} \) be an \( A \)-vector. The \( m \)-outcome of \( \sigma_A \), noted out\((\sigma_A,m)\), is the set of all indexes \( c \) such that \( \sigma = \tau'^{-1}(c, m) \) and \( \sigma_A \subseteq \sigma \).

A labeled joker tree \( \langle T_M, V, Q \rangle \) is constructed out of a concurrent game structure \( \mathcal{M} \) and a formula \( \theta \) as follows, starting from a state \( s \) of \( \mathcal{M} \) (the state of \( \mathcal{M} \) at which one wants to evaluate a given formula:

1. Add the root node \( \varepsilon \) to \( \langle T_M, V, Q \rangle \): \( V(\varepsilon) = L(s) \) and \( Q(\varepsilon) = s \).
2. For each node \( t \in \langle T_M, V, Q \rangle \) with \( Q(t) = s' \), if \( s' \neq \emptyset \) then
   (a) compute the maximum number \( m'_t \) of moves available in \( \mathcal{M} \) at state \( s' \) for any agent: \( m'_t = \max_{a \in \Sigma} d_a(s') \);
   (b) for each \( c \in \{0, \ldots, m'^n - 1\} \), add the successor node \( t.c \) to \( \langle T_M, V, Q \rangle \). Let \( \sigma = \tau'^{-1}(c, km'_t) \). If the transition \( \delta(s', \sigma) = s'' \) exists in \( \mathcal{M} \) then \( V(t.c) = L(s'') \) and \( Q(t.c) = s'' \), else, \( Q(t.c) = \emptyset \) and for each coalition \( A \) appearing on \( \theta \) and \( \sigma_A \) such that \( \sigma_A \subseteq \sigma \), \( V(t.c)(A) = Y \) if \( \sigma_A \in D_A(s') \) and \( V(t.c)(A) = N \) if \( \sigma_A \not\in D_A(s') \).

**Example 3.** Again, let \( \theta_1 = \neg((1) \square p \land (\langle 1, 2 \rangle) \circ p \land \neg((2) \diamond p) \) The concurrent game structure \( \mathcal{M}_1 \) used for this example is given in figure 1.

The formula \( \theta_1 \) contains three coalitions, so a joker vector is a triple such that its first, second and third element correspond to coalition \( \{1\} \), coalition \( \{2\} \) and coalition \( \{1, 2\} \), respectively. Let us start the construction of \( \langle T_{M_1}, V, Q \rangle \) with state \( s_0 \). Then the root node of \( \langle T_{M_1}, V, Q \rangle \) is \( \varepsilon \) with \( V(\varepsilon) = \{p\} \) and \( Q(\varepsilon) = s_0 \). For the node \( \varepsilon \), \( m_\varepsilon = \max(d_1(s_0), d_2(s_0)) = \max(1, 2) = 2 \). Since there are 2 players, node \( \varepsilon \) has \( 2^2 \) successor nodes of in \( \langle T_{M_1}, V, Q \rangle \), as shown by the table:
In this example, the CGS $\mathcal{M}_1$ is such that at node $s_0$ the only possible move for agent 1 is 1, reflected by the mark $N$ as first element of the two joker triples in the table above, corresponding to the two vectors where the coalition $\{1\}$ would choose 1. Agent 2, on the contrary, may play either 0 or 1 at $s_0$, thus we set the second mark to $Y$ in these triples. The situation for the coalition $\{1,2\}$ is such that none of the vectors $(1,0)$ and $(1,1)$ is legal, which explains the third element of the two triples to be $N$. This example should make it clear while the naive approach labeling both the 3rd and the 4th successors of $\varepsilon$ with the unique new propositional symbol $NEN$ would have been incorrect. The complete labeled joker tree $\langle T_{\mathcal{M}_1}, V, Q \rangle$ is represented in Figure 2.

We now show how to modify the tableau procedure so as to keep track of the information about which coalitions are implied in the next-time formulas selected by rule Next. The coalition information is indicated by $[\langle A \rangle]$ for positive next-time formulas and by $[\neg \langle A \rangle]$ for negative next-time formulas. The prestates and the states are now containing also elements referred to as coalition information. Rule SR is also modified in order to pass the coalition information from a prestate to the generated states.

Formally, the only modification of the rule Next is that the prestate $\Gamma_\sigma$ is now defined by:

$$\Gamma_\sigma = \{ \varphi_p, [\langle A_p \rangle] \mid \langle A_p \rangle \circ \varphi_p \in \Delta \text{ and } \sigma_a = p \text{ for all } a \in A_p \}$$

$$\cup \{ \neg \varphi_q, [\neg \langle A'_q \rangle] \mid \neg \langle A'_q \rangle \circ \varphi_q, \neg \text{e}(\sigma) = q, \text{ and } (\Sigma \theta - A'_q) \subseteq N(\sigma) \}$$

The first step of rule SR becomes:

1. add to the pretableau all the minimal downward saturated extensions $\Delta$ of $\Gamma$ as states. If $\Gamma$ contains coalition information elements $[\langle A \rangle]$ or $[\neg \langle A \rangle]$, then add them to $\Delta$.

Beside these minor modifications, the tableau procedure keeps the same. The semi-final tableau $\mathcal{S}T^\theta_1$ for the formula $\theta_1 = \neg \langle 1 \rangle \circ \varphi_p \land \langle 1, 2 \rangle \circ \neg \langle 2 \rangle \circ \neg \varphi_q$ obtained with the new tableau procedure is available in the appendix, figure 5.

The notations $\text{vect}(\Delta, \phi)$ used below have been defined in Section 3, Definition 5. Moreover, we recall that the notations $\Delta_{Am}$ and $\text{out}(\sigma, m)$ used below refer to joker trees and have been introduced in Definition 9. Again, the notation $\text{succ}_\sigma(q)$ refers to tableaux, and means the set of states that are successors of the state $q$ via the move vector $\sigma$ in a semi-final tableau for the formula $\theta$. Again, $\Sigma$ is the set of agents occurring in $\theta$.

**Definition 10.** Given a formula $\theta$, the tableau-based alternating automaton or joker automaton $\mathfrak{A}_\theta$ is defined as the tuple $= (\Theta, \mathcal{D}, Q, I, \rho, F)$ where:

1. $\Theta = \text{AP}$ is the set of atomic propositions in $\theta$. 

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Successors of $\varepsilon$ & $\sigma$ & $V$ & $Q$ \\
\hline
0 & (0,0) & $\{p\}$ & $s_1$ \\
1 & (0,1) & $\{p\}$ & $s_2$ \\
2 & (1,0) & $\{N, Y, N\}$ & $\emptyset$ \\
3 & (1,1) & $\{N, Y, N\}$ & $\emptyset$ \\
\hline
\end{tabular}
\end{table}
\[ \delta(s_0, 0, 0) \]
\[ \delta(s_0, 0, 1) \]
\[ \delta(s_0, 0, 2) \]
\[ \delta(s_1, 0, 0) \]
\[ \delta(s_1, 0, 1) \]
\[ \delta(s_1, 0, 2) \]
\[ \delta(s_2, 0, 0) \]
\[ \delta(s_2, 0, 1) \]
\[ \delta(s_2, 0, 2) \]
\[ \delta(s_3, 0, 0) \]
\[ \delta(s_3, 0, 1) \]
\[ \delta(s_3, 0, 2) \]
\[ \delta(s_4, 0, 0) \]
\[ \delta(s_4, 0, 1) \]
\[ \delta(s_4, 0, 2) \]
\[ \delta(s_5, 0, 0) \]
\[ \delta(s_5, 0, 1) \]
\[ \delta(s_5, 0, 2) \]

Fig. 1. Graphic representation of a concurrent game structure \( M_2 \).

\[ \langle T_{M_1}, V, Q \rangle \]
\[ \theta_1 \]
\[ \neg \langle \langle 1 \rangle \rangle \Box p \land \langle \langle 1, 2 \rangle \rangle \lozenge p \land \neg \langle \langle 2 \rangle \rangle \lozenge p \text{.} \]

Fig. 2. Labeled joker tree \( \langle T_{M_1}, V, Q \rangle \) constructed from the concurrent game structure \( M_1 \) for the ATL formula \( \theta_1 \).
2. \( \mathcal{D} \) is a finite set of arities.

3. \( Q = S^\theta \) is the set of states of a semi-final tableau \( ST^\theta \), excluding the state \( \{ \top, \langle \Sigma \rangle \cup \top \} \).

4. \( I \subseteq Q \) is the set of states containing the ATL-formula \( \theta \).

5. \( F \subseteq Q \) is the set of states whose next-time formulas are either of the form \( \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \square \phi \) or the form \( \neg \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \phi_1 \cup \phi_2 \).

6. The transition function \( \rho \) is defined by:

   (a) for any \( \pi \in 2^\mathcal{D} \) and for any arity \( d \in \mathcal{D} \):
   \[
   \rho(q, \pi, d) = \bigwedge_{\varphi \in \text{PRIM}(q)} \mu(\varphi, \pi, m)
   \]

   where \( \text{PRIM}(q) \) is the set of primitive formulas in \( q \), \( m = e^{(\ln(d)/|\Sigma|)} \) and:
   
   i. \( \mu(p, \pi, k) = \text{true} \) if \( p \in \pi \)
   
   ii. \( \mu(p, \pi, k) = \text{false} \) if \( p \notin \pi \)
   
   iii. \( \mu(\neg p, \pi, k) = \text{true} \) if \( p \notin \pi \)
   
   iv. \( \mu(\neg p, \pi, k) = \text{false} \) if \( p \in \pi \)
   
   v. \( \mu(\top, \pi, k) = \text{true} \)
   
   vi. \( \mu(\langle \Sigma \rangle) \cap \top, \pi, k = \text{true} \)
   
   vii. \( \mu(\langle A \rangle) \circ \varphi', \pi, k) = \bigwedge_{\sigma \in \Delta_{\text{out}}} \left( \bigvee_{c \in \text{out}(\sigma, m)} \left( \bigwedge_{\sigma' \in \text{vect}(\langle \langle A \rangle \rangle \circ \varphi')} \left( \bigvee_{q' \in \text{succ}_{\sigma'}(q)} \left( \mu(q', \varphi', k) \right) \right) \right) \right) \)
   
   viii. \( \mu(\neg \langle \langle A \rangle \rangle) \cup \varphi', \pi, k) = \bigwedge_{\sigma \in \Delta_{\text{out}}} \left( \bigvee_{c \in \text{out}(\sigma, m)} \left( \bigwedge_{\sigma' \in \text{vect}(\langle \langle A \rangle \rangle \circ \varphi')} \left( \bigvee_{q' \in \text{succ}_{\sigma'}(q)} \left( \mu(q', \varphi', k) \right) \right) \right) \right) \)

   (b) for all \( \mathcal{J} \in \{ \mathcal{Y}, \mathcal{N} \}^{nc} \), with \( nc \) the number of coalitions in \( \theta \) by:
   \[
   \rho(q, \mathcal{J}) = \bigwedge_{[C] \in \text{COAL}(q)} \mu([C], \mathcal{J})
   \]

   where \( \text{COAL}(q) \) is the set of coalition information elements of the form \( \langle \langle A \rangle \rangle \) or \( \neg \langle \langle A \rangle \rangle \) in \( q \) and:
   
   i. \( \mu([\langle \Sigma \rangle], \mathcal{J}) = \text{false} \)
   
   ii. \( \mu([\neg \langle \Sigma \rangle], \mathcal{J}) = \text{true} \)
   
   iii. \( \mu([\langle A \rangle], \mathcal{J}) = \text{true} \) if \( \mathcal{J}(A) = \mathcal{Y} \)
   
   iv. \( \mu([\langle A \rangle], \mathcal{J}) = \text{false} \) if \( \mathcal{J}(A) = \mathcal{N} \)
   
   v. \( \mu([\neg \langle A \rangle], \mathcal{J}) = \text{true} \) if \( \mathcal{J}(A) = \mathcal{N} \)
   
   vi. \( \mu([\neg \langle A \rangle], \mathcal{J}) = \text{false} \) if \( \mathcal{J}(A) = \mathcal{Y} \)

Again, an accepting run of this automaton is such that each of its branches, say \( \lambda \), is either finite (because no invariant has been met) and ends with \( \text{true} \) or else is infinite and meets infinitely often a final state, that is a state whose only next-time formulas are of the form \( \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \square \phi \) or \( \neg \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \phi_1 \cup \phi_2 \). Whenever \( \lambda \) meets \( \text{true} \) (and stops) or else meets a final state, this means that no eventuality \( E \) so far encountered is any longer pending, that is, \( E \) has been realized. Intuitively, this means that, given an automaton \( \mathfrak{A}_\theta \), a run reading
Example 4. Automaton $\mathfrak{A}_{\theta_1}$ for $\theta_1 = \neg\langle\langle 1 \rangle\rangle \Box p \land \langle\langle 1, 2 \rangle\rangle \circ p \land \neg\langle\langle 2 \rangle\rangle \circ \neg p$

1. $\Theta = \{p\}$
2. $D = \{1, 4\}$
3. $Q = \{q_1, \ldots, q_6\}$
4. $I = \{q_1, q_2\}$
5. $F = \emptyset$. In this example, a run must be finite with all its leaves equal to $\text{true}$ to be accepted, because no invariant is there.

Below we give a few examples of the 40 transitions:

$$
\rho(q_1, \{p\}, 1) = \mu((\langle\langle 1, 2 \rangle\rangle \circ p, \{p\}), 1) \land \mu(\neg\langle\langle 2 \rangle\rangle \circ \neg p, \{p\}, 1) \land \\
\mu(\neg\langle\langle 1 \rangle\rangle \circ \langle\langle 1 \rangle\rangle \Box p, \{p\}, 1) = (0, q_3) \land (0, q_6) \lor (0, q_3) \lor (0, q_4)
$$

$$
\rho(q_1, \{p\}, 4) = \mu((\langle\langle 1, 2 \rangle\rangle \circ p, \{p\}, 2) \land \mu(\neg\langle\langle 2 \rangle\rangle \circ \neg p, \{p\}, 2) \land \\
\mu(\neg\langle\langle 1 \rangle\rangle \circ \langle\langle 1 \rangle\rangle \Box p, \{p\}, 2)
$$

$$
= (0, q_5) \lor (1, q_5) \lor (2, q_3) \lor (3, q_5) \land ((0, q_6) \lor (2, q_6)) \land \\
(1, q_6) \lor (3, q_6) \land ((0, q_4) \lor (0, q_4) \lor (1, q_4) \lor (1, q_4)) \land \\
(2, q_3) \lor (2, q_4) \lor (3, q_3) \lor (3, q_4))
$$

$$
\rho(q_3, \{Y, N, N\}) = \rho(q_4, \{Y, N, N\}) = \mu(\neg\langle\langle 1 \rangle\rangle), \{Y, N, N\}) = \text{false}
$$

$$
\rho(q_3, \{N, Y, N\}) = \rho(q_4, \{N, Y, N\}) = \mu(\neg\langle\langle 1 \rangle\rangle), \{N, Y, N\}) = \text{true}
$$

A run for this automaton over the labeled joker tree of Figure 2 can be found in appendix, figure 6.

**Theorem 1 (Soundness and Completeness).** If there exists an accepting run $(T_r, r)$, respecting strategies, of the tableau-based alternating automaton $\mathfrak{A}_\theta$ over a labeled joker tree $(T_M, V, Q)$ constructed from a concurrent game structure $M$ and an initial state $s$ of $M$, then $M, s \vdash \theta$. Conversely, if there exists a concurrent game structure $M$, and a state $s_0$ of $M$ such that $M, s_0 \vdash \theta$, then $\mathfrak{A}_\theta$ accepts the labeled joker tree $(T_M, V, Q)$ — constructed from $M$ and $s_0$ — with an accepting run $(T_r, r)$.

The complete proof of this result is given in the next section. Its intuitive meaning is that the joker automaton associated to a formula $\theta$ recognizes exactly the set of all models of $\theta$ (suitably encoded).

## 5 Proofs

### 5.1 Proof of Proposition 1

**Soundness** Here, we prove that the first automaton based on tableau that we have defined, not using jokers and noted $\mathcal{A}_\theta'$, is sound and complete, as stated by Proposition 1.
Here a $k^n$-branching labeled tree $⟨T, V⟩$ is taken to be itself a CGS interpreting θ, where each player has exactly k possible moves at each state and each move consists in choosing one of the $k$ next-time formulas. The model has an infinite number of states because distinct nodes correspond to distinct states. As a consequence, given any coalition $A$, no matter which choice $A$ makes at any state, such a choice will belong to a strategy, since no state can occur in the CGS-tree twice.

First, we prove the soundness of the automaton construction: if there exists an accepting run $⟨T_r, r⟩$ of the automaton $A'_θ$ over a $k^n$-branching labeled tree $⟨T, V⟩$, then $⟨T, V⟩, t ⊨ θ$.

To facilitate the proof, we use the notion of sub-run defined in [7] but modified as follows:

Definition 11. Let $y ∈ T_r$ and $q$ be a state of $A'_θ$ such that $r(y) = (t, q)$. The sub-run $⟨T^y_q, r^y_q⟩$ is the tree with nodes $z ∈ T^y_q$ iff $y.z ∈ T_r$ and $r^y_q(ε) = (t, q), r^y_q(z) = r(y.z)$ for $z ≠ ε$.

Thus $⟨T^y_q, r^y_q⟩$ is the sub-tree of $⟨T_r, r⟩$ taking node $y ∈ T_r$ as root node and modified so that the new root node is labeled by $q$. Note that $⟨T^y_q, r^y_q⟩ = ⟨T_r, r⟩$ when $y = ε$ and $q ∋ θ$.

Also, define $A'_qθ$ to be the automaton $A'_θ$ modified so as to have an initial state $q$.

The sub-runs are defined so as to have the following property: a sub-run $⟨T^y_q, r^y_q⟩$ is an accepting run of the automaton $A'_qθ$.

In order to get the desired result, what we actually prove, by induction on ϕ, is the following “double property”: If $⟨T^y_q, r^y_q⟩$ is accepting, then for all nodes $y ∈ T_r$ where $r(y) = (t, q)$:

$P_+ :$ If $ϕ ∈ q$, then $⟨T, V⟩, t ⊨ ϕ$.

$P_− :$ If $¬ϕ ∈ q$, then $⟨T, V⟩, t ⊭ ϕ$ (i.e. $⟨T, V⟩, t ⊨ ¬ϕ$).

Let us suppose that $⟨T^y_q, r^y_q⟩$ is is accepting.

Base

\[
\varphi = p, \ p \in AP
\]

$P_+ :$ If $ϕ ∈ q$ then $µ(p, V(t))$ holds true, that is $p ∈ V(t)$. Thus $⟨T, V⟩, t ⊨ p$, hence $⟨T, V⟩, t ⊨ ϕ$. Thus $P_+$ holds for $ϕ$.

$P_− :$ If $¬ϕ ∈ q$ then $µ(¬p, V(t))$ holds true, that is $p \notin V(t)$. Thus $⟨T, V⟩, t ⊨ ¬p$, hence $⟨T, V⟩, t ⊨ ¬ϕ$. Thus $P_−$ holds for $ϕ$.

Inductive Step

\[
\varphi = ¬p
\]

$P_+ :$ If $ϕ ∈ q$, then, since $p$ is a proper subformula of $ϕ$, property $P_−$ of $p$ gives that $⟨T, V⟩, t ⊨ ¬p$. Hence, $⟨T, V⟩, t ⊨ ϕ$.

$P_− :$ If $¬ϕ ∈ q$, then by rule $SR$ the atom $p ∈ q$, thus property $P_+$ of $p$ gives
\(\langle T, V \rangle, t \vDash p\), equivalent to \(\langle T, V \rangle, t \not\vDash \neg p\). Hence, \(\langle T, V \rangle, t \not\vDash \varphi\).

\[
\varphi = \varphi_1 \land \varphi_2
\]

\(+\) : If \(\varphi \in q\), then \(\varphi_1 \in q\) and \(\varphi_2 \in q\) by the rule **(SR)** of the tableau procedure. So, by induction hypothesis (\(P+\)), \(\langle T, V \rangle, t \vDash \varphi_1\) and \(\langle T, V \rangle, t \vDash \varphi_2\) respectively. Thus \(\langle T, V \rangle, t \vDash \varphi_1 \land \varphi_2\), hence \(\langle T, V \rangle, t \vDash \varphi\).

\(-\) : If \(\neg \varphi \in q\), then, by the rule **(SR)** of the tableau procedure, either \(\neg \varphi_1 \in q\) or \(\neg \varphi_2 \in q\). In the first case, by induction hypothesis (\(P-\)), \(\langle T, V \rangle, t \vDash \neg \varphi_1\). The second case is similar. Thus \(\langle T, V \rangle, t \vDash \neg (\varphi_1 \land \varphi_2)\) and \(\langle T, V \rangle, t \not\vDash \varphi\).

\[
\varphi = \varphi_1 \lor \varphi_2
\]

This case is similar to the one above.

\[
\varphi = \top
\]

\(+\) : If \(\varphi \in q\) then \(\mu(\top, V(t))\) holds true, that is \(V(t)\) can be any set of \(2^A\). Thus \(\langle T, V \rangle, t \vDash \top\), hence \(\langle T, V \rangle, t \vDash \varphi\). Thus \(P+\) holds for : \(\varphi\).

\(-\) : This case is impossible since \(\langle T_p^{q,r}, r^{y,q} \rangle\) is accepting.

\[
\varphi = (\langle \Sigma \rangle) \bigcirc \top
\]

\(+\) : If \(\varphi \in q\) then \(\mu((\langle \Sigma \rangle) \bigcirc \top, V(t))\) holds true, that is, there exits a successor \(t.c\) such that \(V(t.c)\) can be any set of \(2^A\). So \(\langle T, V \rangle, t \vDash (\langle \Sigma \rangle) \bigcirc \top\), hence \(\langle T, V \rangle, t \vDash \varphi\). Thus \(P+\) holds for : \(\varphi\).

\(-\) : This case is impossible since \(\langle T_p^{q,r}, r^{y,q} \rangle\) is accepting.

\[
\varphi = (\langle A \rangle) \bigcirc \varphi'
\]

\(+\) : If \(\varphi \in q\) then \(\langle T_p^{y,q}, r^{y,q} \rangle\) is accepting implies that there exists a set \(\varOmega\) of pairs \((c, q')\) satisfying:

\[
\forall \sigma \in \Delta_A \left( \bigwedge_{c \in \text{out}(\sigma)} \left( \forall \sigma^* \in \text{vect}(q, \langle A \rangle) \bigcirc \varphi' \right) \left( \forall q' \in \text{succ}_{c^*}(q) \langle c, q' \rangle \right) \right)
\]

Thus, let \(\sigma_A\) be some appropriate vector in \(\Delta_A\) (the set of all the possible A-moves), and let us write elements of \(\text{out}(\sigma_A)\) as \(c_0, \ldots, c_{n-1}\) with \(n = |\text{out}(\sigma_A)|\). For each \(c_i\), \(0 \leq i \leq n - 1\), there are some \(\sigma^* \in \text{vect}(q, \langle A \rangle) \bigcirc \varphi'\) and some \(q_i \in \text{succ}_{c^*}(q)\) such that \((c_i, q_i) \in \varOmega\) and then \(r^{y,q_i}(y, i) = (t.c_i, q_i)\) by run definition. So, for each \((c_i, q_i) \in \varOmega\), \(\varphi' \in q_i\) by definition of \(\text{vect}(q, \langle A \rangle) \bigcirc \varphi'\) (and rule **Next** construction), and \(\langle T_p^{y,q_i}, r^{y,q_i} \rangle\) is accepting, since all sub-runs of \(\langle T_p^{y,q}, r^{y,q} \rangle\) are accepting. By induction hypothesis, using the property \(P+\) of \(\varphi'\), we get \(\langle T, V \rangle, t.c_i \vDash \varphi'\) for some \(\sigma \in \text{varDelta}_A\) and all \(t.c_i \in \text{out}(\sigma_A)\). Thus \(\langle T, V \rangle, t \vDash (\langle A \rangle) \bigcirc \varphi'\), hence \(\langle T, V \rangle, t \vDash \varphi\) and hence \(P+\) holds for \(\varphi\).

\(-\) : If \(\neg \varphi \in q\), that is \(\neg(\langle A \rangle) \bigcirc \varphi'\) where \(A \not\in \Sigma\) then \(\langle T_p^{y,q}, r^{y,q} \rangle\) is accepting implies that there exists a set \(\varOmega\) of pairs \((c, q')\) satisfying:

\[
\forall \sigma \in \Delta_A \left( \bigwedge_{c \in \text{out}(\sigma)} \left( \forall \sigma^* \in \text{vect}(q, \neg(\langle A \rangle) \bigcirc \varphi') \left( \forall q' \in \text{succ}_{c^*}(q) \langle c, q' \rangle \right) \right) \right)
\]

Let \(m = |\Delta_A|\) and let \(\sigma_0, \ldots, \sigma_{m-1}\) be all the elements of \(\Delta_A\). For each \(\sigma_i\), \(0 \leq i \leq m - 1\), there are some \(c_i \in \text{out}(\sigma_i)\) and also some \(\sigma^* \in \text{vect}(q, \neg(\langle A \rangle) \bigcirc \varphi')\) and
some \( q_i \in \text{succ}_{\tau}(q) \) such that \((c_i, q_i) \in \Omega\), and then \( r^{y,q}(y, i) = (t.c_i, q_i) \) by run definition. So, for each \((c_i, q_i) \in (Q), \neg \varphi' \in q_i\), by definition of vect\((q, \langle A \rangle)\circ \varphi'\) and of the (\textbf{Next}) rule of the tableaux, and \( (T_{x:i}^{y,q}, r^{y,q}) \) is accepting since all sub-runs of \( (T_{x:i}^{y,q}, r^{y,q}) \) are accepting. By induction hypothesis, for all \( \sigma \in \Delta_A \) (that is the set of all the possible \( A \)-moves at \( t \)), there exists some \( t.c_i \in \text{out}(\sigma) \) such that \( \langle T, V \rangle, t.c_i \vdash \neg \varphi' \). Thus \( \langle T, V \rangle, t \vdash \neg \varphi \) and \( P^- \) holds for \( \varphi \).

\[ \varphi = \neg \langle \langle A \rangle \rangle \circ \varphi' \]

+ : This case reduces to the proof of \( P^- \) for \( \langle \langle A \rangle \rangle \circ \varphi' \) (see above).

- : If \( \neg \langle \langle A \rangle \rangle \circ \varphi' \in q \), by rule \textbf{SR} the equivalent formula \( \langle \langle A \rangle \rangle \circ \varphi \) belongs to \( q \). Thus, the proof of \( P^- \) for this case reduces to the proof of \( P^+ \) for \( \langle \langle A \rangle \rangle \circ \varphi \) (see above).

\[ \varphi = \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \]

(is this an \textit{eventuality})

+ : If \( \varphi \in q \) then \( \varphi_1 \in q \) and \( \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \in q \), or else \( \varphi_2 \in q \), by the rule (\textbf{SR}). In the latter case, we can apply the induction hypothesis \( (P^+ \) to \( \varphi_2 \) so as to obtain \( \langle T, V \rangle, t \models \varphi_2 \), which obviously implies \( \langle T, V \rangle, t \models \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \) and we are done. In the other case, we can indeed apply the induction hypothesis \( (P^+ \) to \( \langle T, V \rangle, t \models \varphi_1 \), but \textit{we cannot apply the induction hypothesis to get} \( \langle T, V \rangle, t \models \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \), because \( \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \) is not a subformula of \( \varphi \). Thus, we must reason differently, for this case. By hypothesis the run \( \langle T_{x:i}^{y,q}, r^{y,q} \rangle \) is accepting and this means that each of its branches, say \( \beta \), is either finite and ends with true or else is infinite and meets infinitely often a final state of the automaton. Since we are in the case where \( \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \in q \), this means that \( \beta \) has a segment of the form \( y = n_0, n_1, \ldots, n_{p-1}, q_p \) where \( r(y) = r(n_0) = (t, q), r(n_1) = (t_1, q_1), \ldots, r(n_{p-1}) = (t_{p-1}, q_{p-1}), r(n_{p-1}) = (t_p, q_p) \) and:

1. For each \( 1 \leq j \leq p \), \( n_j \) is a successor in the run tree of the node \( y = n_{j-1} \) and \( t_j \) is the CGS state obtained by a move \( \sigma_j \) of coalition \( A \) at the corresponding state \( t_{j-1} \) of the CGS (i.e. the labeled tree \( \langle T, V \rangle \));
2. For each \( 1 \leq j \leq p \), \( q_j \) is a state of the automaton containing \( \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \);
3. For each \( 1 \leq j < p \), \( \varphi_1 \in q_j \);
4. \( \varphi_2 \in q_p \).

In fact, whenever a branch meets an accepting state this means that \textit{each eventuality so far encountered has been solved}. Each sub-run of \( \langle T_{x:i}^{y,q}, r^{y,q} \rangle \) rooted at \( n_i \) is accepting, therefore by induction hypothesis \( (P^+ \) for each \( j \) where \( 1 \leq j < p \), \( \langle T, V \rangle, t_j \models \varphi_1 \) and \( \langle T, V \rangle, t_p \models \varphi_2 \). Since each node of the tree \( \langle T, V \rangle \) represents a different state of the interpretation, any behavior of the coalition \( A \) such that \( \sigma_1 \) is \( A \)'s move at state \( t \) and, for each \( j \) with \( 1 \leq j \leq p \), \( \sigma_j \) is \( A \)'s move at state \( t_{j-1} \) allows one to conclude that \( \langle T, V \rangle, t_j \models \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \) and, finally, that \( \langle T, V \rangle, t \models \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \). That is, \( P^+ \) holds for \( \varphi \).

- : If \( \neg \varphi \in q \), this means that \( q \) contains \( \neg \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \) (an invariant). Therefore \( \neg \varphi_1 \in q \) and \( \neg \varphi_2 \in q \), or else \( \neg \varphi_2 \in q \) and \( \langle \langle A \rangle \rangle \circ \langle \langle A \rangle \rangle \varphi_1 U \varphi_2 \in q \), by
the rule SR. In the first case, we can apply the induction hypothesis (P−) to ϕ1 and ϕ2 so as to obtain \( (T, V), t \models \neg \varphi_1 \) and \( (T, V), t \models \neg \varphi_2 \), which obviously implies \( (T, V), t \models \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \) and we are done. In the other case, we can indeed apply the induction hypothesis (P−) to get \( (T, V), t \models \neg \varphi_2 \), but we cannot apply the induction hypothesis to get \( (T, V), t \models \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \).

By hypothesis the run \((T^n_y, r^n_y)\) is accepting and this means that each of its branches, say β, is either finite and ends with true or else is infinite and meets infinitely often a final state of the automaton. Since we are in the case where 
\[ \neg \langle \langle A \rangle \rangle \bigcirc \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \in q, \]
this means that β has the form \( y = n_0, n_1, n_2, \ldots \) where \( r(y) = r(n_0) = (t, q), r(n_1) = (t_1, q_1), r(n_2) = (t_2, q_2) \ldots \) and:

1. For each \( j \geq 1, n_j \) is a successor in the run tree of the node \( y = n_{j-1} \) and \( t_j \) is the CGS state obtained by a co-move \( \sigma_j \) of coalition A at the corresponding state \( t_{j-1} \) of the CGS (i.e. the labeled tree \( (T, V) \));

2. For each \( j \geq 1, q_j \) is a state of the automaton containing \( \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \);

3. For each \( j \geq 1, \neg \varphi_2 \in q_j \).

Each sub-run of \((T^n_y, r^n_y)\) rooted at \( n_i \) is accepting, therefore by induction hypothesis (P−) for each \( j \) where \( j \geq 1 \), \( (T, V), t_j \models \neg \varphi_2 \) and since each node of the tree \((T, V)\) represents a different state of the interpretation, we can conclude that \( (T, V), t_j \models \neg \langle \langle A \rangle \rangle \bigcirc \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \) and, finally, that \( (T, V), t \models \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \).

That is, \( P− \) holds for \( \varphi \)

\[ \varphi = \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \]

+ : This case reduces to the proof of \( P− \) for \( \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \) (see above).

− : if \( \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \in q \), by rule SR the equivalent formula \( \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \) belongs to \( q \). Thus, the proof of \( P− \) for this case reduces to the proof of \( P+ \) for \( \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \) (see above).

\[ \varphi = \langle \langle A \rangle \rangle \Box \varphi_1 \]

Formulas of type \( \varphi = \langle \langle A \rangle \rangle \Box \varphi_1 \) are invariant and the proofs for \( P+ \) and \( P− \) are similar to the case \( \neg \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \).

\[ \varphi = \neg \langle \langle A \rangle \rangle \Box \varphi_1 \]

Formulas of type \( \varphi = \neg \langle \langle A \rangle \rangle \Box \varphi_1 \) are eventuality and the proofs for \( P+ \) and \( P− \) are similar to the case \( \langle \langle A \rangle \rangle \varphi_1 \varphi_2 \).

All the other cases of \( \varphi \) can be reduced to the above ones.

**Completeness** Below, we prove the completeness of the automaton construction, that is:

if there exists a \( k^n \)-labeled tree \((T, V)\) such that \( (T, V), \varepsilon \vDash \theta \), then \( A'_\theta \) accepts \( (T, V) \) with an accepting run \( (T_r, r) \).

The fact that the automaton states are tableau states of the semi-final tableau for \( \theta \) is central to this proof, because this allows us to use properties of tableaux.
Let us recall that if $q$ is a set of formulas (a state of the tableau, actually), the notation $M, s \models q$ to means that, for every formula $\psi \in q$, $M, s \models \psi$

Thus, let us suppose that there exists a $k^n$-labeled tree $\langle T, V \rangle$ such that $\langle T, V \rangle, \varepsilon \models \theta$ (once again, here the tree $\langle T, V \rangle$ is taken to be an infinite CGS).

The construction of the run inductively adds nodes so as to ensure that for all new nodes $y \in T_r$, if $r(y) = (t, q)$ then $\langle T, V \rangle, t \models \theta$. Nodes to be added are chosen non-deterministically, so as that when the procedure is determinized, what is actually built is a class of runs associated to $A'$ and the $k^n$-labeled tree $\langle T, V \rangle$.

**Initialization** The initialization of the construction creates the root node $\varepsilon$ of $\langle T_r, r \rangle$. The label of the root must contain an initial state of the automaton, that is a state of the semi-final tableau for $\theta$ that is a downward saturation of $\theta$. In general, several choices are possible, among the states obtained in the tableau from the initial prestate $\{\theta\}$ of the semi-final tableau. Thus, let us non-deterministically choose one state $q_{\text{init}}$ among such initial states. Certainly, at least one initial state will be such that $\langle T, V \rangle, t \models q_{\text{init}}$, because of the soundness of the $\text{SR}$ rule and the fact that $\langle T, V \rangle, \varepsilon \models \theta$. It also worthwhile noticing that the semi-final tableau obtained via the reduced pruning procedure of $\text{P}^\theta$ excluding the application of the rule $\text{E}3$ contains as many satisfiable states as the final tableau pruned by using also $\text{E}3$, because, as shown in [6], no satisfiable state of a semi-final tableau will be eliminated on the account of $\text{E}3$ (see p.31). Hence we can set as label $r$ of the initial state of the run, the couple $(\varepsilon, q_{\text{init}})$.

**Inductive step** Now, let us consider any newly added node $y \in T_r$, with $r(y) = (t, q)$ and $\langle T, V \rangle, t \models q$. The objective is to construct a set $\Omega = \{(c_1, q_1), \ldots, (c_n, q_n)\}$ satisfying $\rho(q, V(t))$ such that for all $1 \leq i \leq n$, $y.i \in T_r$, $r(y.i) = (t, c_i, q_i)$ and $\langle T, V \rangle, t, c_i \models q_i$. The set $\Omega$ is constructed from all primitive formulas $\varphi$ of state $q$.

In order to create successors for the node $r(y) = (t, q)$, we need to determine two things: the nodes that the automaton will read and the states used to read them.

Before that, let $\{(\langle A_0 \rangle) \circ \varphi_0, \ldots, (\langle A_{m-1} \rangle) \circ \varphi_{m-1}, (\langle \emptyset \rangle) \circ \varphi_0', \ldots, (\langle \emptyset \rangle) \circ \varphi_{s-1}'\}$ be the set of primitive formulas in $q$. Such a set can be partitioned in $m + l$ subsets, say $q_0, \ldots, q_{m-1}$ and $q_0', \ldots, q_{s-1}'$, where each $q_i$ will be such that $q_i = \{(\langle A_i \rangle) \circ \varphi, (\langle \emptyset \rangle) \circ \varphi_0', \ldots, (\langle \emptyset \rangle) \circ \varphi_s'\}$ where $A_i \neq \emptyset$ and $q_i' = \{\neg (\langle A_i' \rangle) \circ \varphi, (\langle \emptyset \rangle) \circ \varphi_0', \ldots, (\langle \emptyset \rangle) \circ \varphi_{s-1}'\}$ where $A_i' \neq \Sigma$.

For each $q_i$, we can find in the pretableau of $\theta$, a prestate, say $\Gamma_i$, equal to $\{\varphi, \varphi_0, \ldots, \varphi_{s-1}\}$. This prestate is the successor of $\overline{q}$ with the move vector corresponding to the case where all agents choose the formulas $\langle A_i \rangle \circ \varphi_i$. Then we choose in $\langle T, V \rangle$ a $A_i$-move $\sigma_{A_i} \in \Delta_{A_i}$ such that for all $c \in \text{out}(\sigma_{A_i}) = \{c_1, \ldots, c_n\}$, $\langle T, V \rangle, t, c \models \Gamma_i$. Such a $A_i$-move $\sigma_{A_i}$ exists, since $\langle T, V \rangle, t \models q$. Thus, in the considered semi-final tableau for $\theta$ that we are considering, for each $c \in \{c_1, \ldots, c_n\}$ there exist at least one element of $\text{States}(\Gamma_i)$, say $s_{ti}$, such that $\langle T, V \rangle, t, c \models s_{ti}$. Such a state exists otherwise $q$ would have been eliminated by
rule \textbf{E2}. Thus for each \( q_i \) and for each \( c \in \text{out}(\sigma_{A_i}) \) the couple \((c, st_i)\) is an element of \( \Omega \).

For each \( q'_i \), we can find in the pretableau of \( \theta \), a prestate, say \( \Gamma'_i \), equal to \( \{ \neg \psi_i, \varphi_0, \ldots, \varphi_{s-1} \} \). This prestate is the successor of \( q \) with a move vector \( \sigma \) corresponding to the case where all agents choose a negative formula and \( \text{neg}(\sigma) = i \). Then for each \( A_i \)-move \( \sigma_{A_{ij}} \in \Delta_{A_i} \) we choose in the pretableau of \( \theta \), a prestate, say \( \Gamma'_{ij} \), such that \( \Gamma'_{ij} = \{ \neg \psi_i, \varphi_0, \ldots, \varphi_{s-1} \} \). This prestate is the successor of \( q \) with a move vector \( \sigma_{A_{ij}} \) corresponding to the case where all agents choose a negative formula and \( \text{neg}(\sigma) = i \).

At the end of the construction, \( \Omega \) is a required set that satisfies

\[
\rho(q, V(t)) = \bigwedge_{\varphi \in \text{run}(q)} \mu(\varphi, V(t))
\]

and is such that its finite branches end with true.

Successors of \( y \) in \( (T_r, r) \) are all \( y.i \) with \( 0 \leq i \leq |\Omega| \) and \( r(y,i) = (c_i, q_i) \in \Omega \).

According to the case, several \( A_i \)-moves \( \sigma_{A_{ij}} \in \Delta_{A_i} \) and several vectors \( c_j \in \Delta_{A_i} \) may be candidates for construction of \( \Omega \) when \( \varphi = \langle \langle A \rangle \rangle \varphi' \) and \( \varphi = \neg \langle \langle A \rangle \rangle \varphi' \), respectively. For each possibility, a different set \( \Omega \) may be obtained and therefore produces a different run \( (T_r, r) \). Let \( \Omega \) be the set of all possible runs \( (T_r, r) \) constructed as above — only runs of \( \Omega \) can be accepting for \( A'_\theta \). Runs of \( \Omega \) can be of two types: \textit{finite} or \textit{infinite}.

- If a run of \( \Omega \) is finite, then all its branches terminate with true and the run is accepting for \( A'_\theta \). This case corresponds to runs for formulas without invariants and realizing all eventualities, if any.
- If a run of \( \Omega \) is infinite then at least one of its branches is infinite, by König Lemma. This case corresponds to runs for formulas with invariant or not realizing all eventualities or both. The automaton only accepts infinite runs, with or without invariants, satisfying all eventualities.

Let us assume that the set \( \Omega \) does not contains any run accepting \( (T_M, V, Q) \), then all runs of \( \Omega \) are infinite and have some branch not satisfying some eventuality. This means that all runs of \( \Omega \) follow a strategy in \( (T, V) \) which does not satisfy an eventuality. Therefore, no strategies in \( (T, V) \) can realize all the eventualities of \( \theta \), otherwise at least one run would have found such a strategy. Thus \( (T, V) \notin \theta \), in contradiction with our hypothesis that \( (T, V) \models \theta \). We can conclude that there exists at least one run \( (T_r, r) \) of \( \Omega \) which is accepted by \( A'_\theta \).

### 5.2 Proof of Theorem 1

The proofs of the previous section carry over, besides details illustrated in the sequel, to the case of the joker automaton that is the core of our contribution, that is to the proof of Theorem 1.
The really major difference is that now an input tree $T$ is not seen as a model itself, but rather as coding any model $M$ (just in the limit case, the model will be $T$ itself). Thus, if $M$ has a finite number of states – that is, it is finite model –, $M$ might have one or several states, even all, whose representation is repeated infinitely often in the input tree. In this case, care must be taken that all the copies in the tree $T$, say $c_1, c_2, \ldots$ of a given state $s$ of $M$ are really treated as representing the same state of the coded CGS $M$. This means that a true strategy of a coalition $A$, while evaluating a given formula $\langle\langle A \rangle\rangle \varphi$ must choose the same move for every $c_i$. Thus a preliminary lemma is needed, showing that the existence of an accepting run not respecting a strategy for $A$ implies the existence of another run respecting it.

Let $\varphi$ be a formula of the form $\langle\langle A \rangle\rangle \varphi_1 U \varphi_2$ or $\langle\langle A \rangle\rangle \square \varphi_1$ and $\Psi\Box$ the next-time formula related to $\varphi$, that is $\Psi\Box = \langle\langle A \rangle\rangle \bigcirc \langle\langle A \rangle\rangle \varphi$. Let $T$ be an accepting run of $\mathcal{A}_0$ over a labelled (joker) tree $\langle T_M, V, Q \rangle$.

**Definition 12.** $T$ is not respecting the strategy for a formula $\varphi$ if there exists two nodes $t, t'$ of $T$ such that

- $t$ is a descendant of $t'$;
- $Q(t) = Q(t')$ that is both $t$ and $t'$ represent a same state of the model $M$ treated by $T$;
- $t$ and $t'$ are associated to an automaton state containing the same next-time formula $\Psi\Box$ as described above;
- the successor nodes of $t$ and $t'$ selected by the automaton when evaluating $\mu(\Psi\Box, \pi, k)$ are not the same. Note that as $Q(t) = Q(t')$, then $\pi$ and $k$ are the same for $t$ and $t'$.

**Definition 13.** We define $M(T)$ as the number of pairs of nodes $(t, t')$ in $T$ which correspond to the description in definition 12.

**Lemma 1.** Let $T$ be an accepting run of $\mathcal{A}_0$ over $\langle T_M, V, Q \rangle$ not respecting the strategy for a formula $\varphi$. Then there exists an accepting run $T'$ of $\mathcal{A}_0$ over $\langle T_M, V, Q \rangle$ respecting the strategy for $\varphi$.

**Proof**

We construct a possibly infinite sequence of accepting runs of $\mathcal{A}_0$ over $\langle T_M, V, Q \rangle$:

$$T = T_0, T_1, T_2, \ldots$$

where, for each $i \geq 0$, if $M(T_i) = 0$ then $T_{i+1} = T_i$, otherwise $M(T_{i+1}) < M(T_i)$. Then, the desired $T'$ will be the limit of such a sequence for $i \to \infty$.

So, let $T_i$ be such $M(T_i) \geq 1$. We can decrease $M(T_i)$ by defining the new run $T_{i+1}$ by means of a modification of the successor nodes selected by the automaton at node $t$. Indeed, the Boolean formulae resulting of the evaluation of $\mu(\Psi\Box, \pi, k)$ at node $t$ and $t'$ are exactly the same, hence it is possible to select the same nodes for $t$ as those for $t'$ and therefore obtain a new run $T_{i+1}$ of $\mathcal{A}_0$ over $\langle T_M, V, Q \rangle$ with $M(T_{i+1}) = M(T_i) - 1$. 
The new run differs from $T_i$ only by the sub-run $(T)$ which root is $t$ and which deals with the evaluation of the sub-transition $\mu(\Psi_{\geq}, \pi, k)$. The rest of $T_i$ remains the same, so all branches from $T \setminus T$ are accepting branches because $T_i$ is an accepting run and all sub-runs of an accepting run are accepting.

Now, we have to prove that the sub-run, say $T'$, replacing $T$ is accepting as well. At node $t$ in run $T_{i+1}$, we apply the same choices of node $t'$, and this means that the behavior of the run from $t$ is now the same than the one at node $t'$ in run $T_i$. As $T_i$ is accepting, all its sub-trees are accepting. So $T$ corresponds to the sub-run $(T')$ in $T_i$ which root is $t'$ and which deals with the sub-transition $\mu(\Psi_{\geq}, \pi, k)$. So $T = T'$. As $T'$ is accepting, then $T$ is also accepting. Thus $T_{i+1}$ is an accepting run of $\mathcal{A}_0$ over $(T_M, V, Q)$ and $M(T_{i+1}) < M(T_i)$.

We can make the similar reasoning for negative formula of the form $\neg \langle \langle (A) \rangle \rangle \varphi_1 \cup \varphi_2$ and $\neg \langle \langle (A) \rangle \rangle \Box \varphi_1$. □

Before detailing the proof, we also make some remarks about joker nodes:

Remark 1. Given a node $t \neq \varepsilon \in (T_M, V, Q)$, $father(t)$ denote the father node of $t$.

(1) Let $t$ be a node of $T_M$. If $\psi \in q$ is of the form $\langle \langle (A) \rangle \rangle$, $A \subseteq \Sigma$, $V(t) \in \{Y, N\}^m$, with $m$ the number of coalitions in $\theta$ and $\langle T_{\Psi,q}, r_{\Psi,q} \rangle$ is an accepting sub-run then $\mu(\langle \langle (A) \rangle \rangle), V(t)) = true$. Thus there exists in $\langle T_M, V, Q \rangle$ some other node $father(t).c \neq t$ with the same $A$-vector as $t$ such that $father(t).c$ is associated to a state in $M$.

(2) If $\psi \in q$, $\psi$ is of the form $\neg \langle \langle (A) \rangle \rangle$, $A \subseteq \Sigma$, $V(t) \in \{Y, N\}^m$, with $m$ the number of coalitions in $\theta$ and $\langle T_{\Psi,q}, r_{\Psi,q} \rangle$ is an accepting sub-run then $\mu(\neg \langle \langle (A) \rangle \rangle), V(t)) = true$. Thus there does not exist in $\langle T_M, V, Q \rangle$ some other node $father(t).c \neq t$ with the same $A$-vector as $t$ such that $father(t).c$ is associated to a state in $M$.

**Soundness**

Here, we want to prove that if there exists an accepting run $\langle T_r, r \rangle$ of the joker automaton $A_0$ over a $k^n$-labeled joker tree $(T_M, V, Q)$ constructed from a CGS $M$ and an initial state $s$ of $M$, then $M, s \models \theta$.

In order to get the desired result, what we actually prove, by induction on $\varphi$, is the following “double property”: If $\langle T_r^n, r^n \rangle$ is accepting, then for all nodes $y \in T_r$ where $r(y) = (t, q)$:

- $P+$ : If $\varphi \in q$, then $M, Q(t) \models \varphi$.
- $P-$ : If $\neg \varphi \in q$, then $M, Q(t) \not\models \varphi$ (i.e. $M, Q(t) \models \neg \varphi$).

This soundness proof is similar to the one done for soundness of proposition 1 and we will just detail the points which differ.

**Base**

- $\varphi = p$, $p \in AP$
  - $P+$ : If $\varphi \in q$ then $\mu(p, V(t))$ holds true, that is $p \in V(t)$ and by construction of $(T_M, V, Q)$, $p \in L(Q(t))$. Thus $M, Q(t) \models p$, hence $M, Q(t) \models \varphi$. Thus $P+$ holds
Moreover, there is a bijective relation \( \varphi \) and \( \psi \) such that \( \neg \varphi \) implies that there exists a set \( Q \). Thus \( \mathcal{M}, Q(t) \models \neg \varphi \), and hence \( \mathcal{M}, Q(t) \models \varphi \). Thus \( P^+ \) holds for : \( \varphi \).

**Inductive Step**

\[ \varphi = (\langle \Sigma \rangle) \circ \top \]

\[ \varphi = (\langle A \rangle) \circ \varphi' \]

\[ \varphi = (\langle \Sigma \rangle) \circ T \]

\[ \varphi = (\langle A \rangle) \circ \varphi' \]

\[ \varphi = (\langle \Sigma \rangle) \circ T \]

\[ \varphi = (\langle A \rangle) \circ \varphi' \]

Thus, let \( \sigma_A \) be some appropriate vector in \( \Delta_{A_m} \) (the set of all the possible \( A \)-moves), and let us write elements of \( \text{out}(\sigma_A, m) \) as \( c_0, \ldots, c_{n-1} \) with \( n = |\text{out}(\sigma_A, m)| \). For each \( c_i, 0 \leq i \leq n-1 \), there are some \( \sigma^* \in \text{vect}(q, \langle A \rangle) \circ \varphi' \) and some \( q_i \in \text{succ}_{\sigma^*}(q) \) such that \( (c_i, q_i) \in \Omega \), and \( r^q(q_i) = (t.c_i, q_i) \) by run definition. So, for each \( (c_i, q_i) \in \Omega \), \( \varphi' \in q_i \), and \( \langle (\langle A \rangle) \circ q_i \rangle \in q_i \) by definition of \( \text{vect}(q, \langle A \rangle) \circ \varphi' \) (and rule **Next** construction), and \( \langle T^q(q_i), r^q(q_i) \rangle \) is accepting, since all sub-runs of \( \langle T^q(q_i), r^q(q_i) \rangle \) are accepting. For some \( \sigma \in \Delta_{A_m} \) and all \( t.c_i \in \text{out}(\sigma_A, m) \), two cases are possible:

1. if \( V(t.c_i) \in 2^{AP} \) then by induction hypothesis, using the property \( P^+ \) of \( \varphi' \), we get \( \mathcal{M}, Q(t.c_i) \models \varphi' \)

2. if \( V(t.c_i) \in \{Y, N\}^{mc} \), then by remark 1 (1), there are some other nodes \( t.c' \neq t.c_i \) with the same \( A \)-vector such that \( Q(t.c') \neq \emptyset \).

Moreover, there is a bijective relation \( s' \leftrightarrow t.c_i \) between all states \( s' \) with \( s' \in M \) and \( s' \in \text{out}(Q(t), \sigma_A) \) and all nodes \( V(t.c_i) \in 2^{AP} \). So there exists some \( \sigma_A \) in \( \mathcal{M} \) at state \( Q(t) \) such that for all \( s' \in \text{out}(Q(t), \sigma_A) \), \( \mathcal{M}, Q(s') \models \varphi' \). Thus \( \mathcal{M}, Q(t) \models (\langle A \rangle) \circ \varphi' \), hence \( \mathcal{M}, Q(t) \equiv \varphi \) and hence \( P^+ \) holds for \( \varphi \).

\[ \varphi \in \mathcal{Q} \mbox{ is accepting implies that there exists a set } \Omega \mbox{ of pairs } (c, q') \mbox{ satisfying:} \]

\[ \bigwedge_{\sigma \in \Delta_{A_m}} \left( \bigvee_{c \in \text{out}(\sigma, m)} \left( \bigvee_{\sigma^* \in \text{vect}(q, (\langle A \rangle) \circ \varphi')} \left( \bigvee_{q' \in \text{succ}_{\sigma^*}(q)} (c, q') \right) \right) \right) \]
Let \( l = |\Delta_{Am}| \) and let \( \sigma_0, \ldots, \sigma_{l-1} \) be all the elements of \( \Delta_{Am} \). For each \( \sigma_i, \ 0 \leq i \leq l-1 \), there are some \( c_i \in \text{out}(\sigma_i) \) and also some \( \sigma^* \in \text{vec}(q,-\langle\langle A\rangle\rangle\circ \varphi') \) and some \( q_i \in \text{succ}_{\sigma^*}(q) \) such that \( (c_i,q_i) \in \Omega \), and then \( r^{q,q}(y,i) = (t.c_i,q_i) \) by run definition. So, for each \( (c_i,q_i) \in \langle Q \rangle \) with \( \neg \varphi' \in q_i \), by definition of \( \text{vec}(q,-\langle\langle A\rangle\rangle\circ \varphi') \) and of the (Next) rule of the tableaux, and \( \langle T^{y,q},r^{y,q} \rangle \) is accepting since all sub-runs of \( \langle T^{q,q},r^{q,q} \rangle \) are accepting. For all \( \sigma \in \Delta_{Am} \) (that is the set of all the possible \( \sigma \)-moves at \( t \)), there exists some \( t.c_i \in \text{out}(\sigma) \) such that.

1. if \( V(t.c_i) \in 2A^P \) then by induction hypothesis, using the property \( P^- \) of \( \varphi' \), we get \( M, Q(t.c_i) \models \neg \varphi' \).
2. if \( V(t.c_i) \in \{Y,N\}^{\text{w}c} \), then by remark 1.2, there are no other nodes \( t.c' \neq t.c_i \) with the same \( \sigma \)-vector such that \( Q(t.c') \neq 0 \).

Then, for all \( \sigma \)-moves \( \sigma_A \) that exists in \( M \) at state \( Q(t) \), all nodes \( t.c' \) with the same \( \sigma \) bijectively corresponds to a sates \( s' \) in \( M \) such that \( s' \in \text{out}(Q(t),\sigma_A) \), \( M, Q(s') \models \neg \varphi' \). Thus \( M, Q(t) \models \neg \langle\langle A\rangle\rangle\circ \varphi' \), hence \( M, Q(t) \models \neg \varphi \) and hence \( P^- \) holds for \( \varphi \).

**Completeness** Our proof of the completeness, consists, first, in establishing a lemma whose proof closely reminds the completeness argument for Proposition 1, while, however, constructing an accepting run that does not necessarily respect strategies.

**Lemma 2.** If there exists a CGS \( M \) and a state \( s_0 \) of \( M \) of such that \( M, s_0 \models \theta \), then \( \mathcal{A}_\theta \) accepts the labeled joker tree \( \langle T_M,V,Q \rangle \) constructed from \( M \) and \( s_0 \) with an accepting run \( \langle T_r,r \rangle \), that, however, might not respect strategies.

**Proof**

Thus, let us suppose that there exists a CGS \( M \) and a state \( s_0 \) of \( M \) such that \( M, s_0 \models \theta \).

Once again, the construction of the run \( \langle T_r,r \rangle \) is inductive and non-determinist, but any newly added node \( y \) \( \langle T_r,r \rangle \) now will be such that if \( r(y) = (t,q) \) then \( M, Q(t) \models q \) if \( Q(t) \neq 0 \) and \( p(q,V(t)) = \text{true} \) if \( Q(t) = 0 \).

**Initialization** The initialization of the construction creates the root node \( \varepsilon \) of \( \langle T_r,r \rangle \). The label of the root must contain an initial state of the automaton, that is a state of the semi-final tableau for \( \theta \) that is a downward saturation of \( \theta \). Several choices being possible, among the states obtained in the tableau from the initial prestate \( \{\theta\} \) of the semi-final tableau, let us non-deterministically choose one state \( q_{\text{init}} \) among such initial states. Certainly, at least one initial state will be such that \( \langle T,V \rangle, t \models q_{\text{init}} \), because of the soundness of the SR rule and the fact that \( \langle T,V \rangle, \varepsilon \models \theta \). Again, we may observe that the semi-final tableau obtained via the reduced pruning procedure of \( P^\theta \) contains as many satisfiable states as the final tableau pruned by using also \( E3 \), because, as shown in [6], no satisfiable state of a semi-final tableau will be eliminated on the account of \( E3 \).
(see p.31). Hence we can set as label \( r(ε) \), the initial state of the run, the couple \((ε, q_{ini})\).

**Inductive step** Now, let us consider any newly added node \( y \in T_r \), with \( r(y) = (t, q) \), \( M, Q(t) \models q \) and \( Q(t) \neq \emptyset \) — joker nodes do not produce new nodes. The objective is to construct a set \( Ω = \{(c_1, q_1), \ldots, (c_n, q_n)\} \) satisfying \( ρ(q, V(t), d(t)) \) such that for all \( 1 ≤ i ≤ n \), \( y.i \in T_r \), \( r(y.i) = (t.c_i, q_i) \) and:

- \( M, Q(t.c_i) \models q_i \) if \( Q(t.c_i) \neq \emptyset \),
- \( ρ(q_i, V(t.c_i), d(t.c_i)) = true \) if \( Q(t.c_i) = \emptyset \).

In order to create successors for the node \( r(y) = (t, q) \), we need to determine two things: the nodes that the automaton will read and the states used to read them.

Before that, let \( \{(A_0) \circ \varphi_0, \ldots, (A_{m-1}) \circ \varphi_{m-1}, (\emptyset) \circ \varphi_0, \ldots, (\emptyset) \circ \varphi_{m-1}\} \) be the set of primitive formulas in \( q \).

Such a set can be partitioned in \( m + l \) subsets, say \( q_0, \ldots, q_{m-1} \) and \( q'_0, \ldots, q'_{l-1} \), where each \( q_i \) will be such that \( q_i = \{(A_{i-1}) \circ \varphi_1, (\emptyset) \circ \varphi'_0, \ldots, (\emptyset) \circ \varphi'_{i-1}\} \) where \( A_1 \neq \emptyset \), and \( q'_0 = \neg((A'_{i_j}) \circ \varphi_2, (\emptyset) \circ \varphi'_0, \ldots, (\emptyset) \circ \varphi'_{l-1}) \) where \( A'_{i_j} \neq \Sigma \).

First, let us reason on the treatment of the \( q_i \), containing only positive next-time formulas.

For each \( q_i \), we can find in the pretableau of \( θ \), a prestate, say \( Π_i \), equal to \( \{\varphi, \varphi_0, \ldots, \varphi_{i-1}\} \). This prestate is the successor of \( q \) with the move vector corresponding to the case where all agents choose the formulas \( (A_i) \circ \varphi_i \).

Then we choose in \( (T_M, V, Q) \) a \( A_i \)-move \( σ_{A_i} \in Δ_{A_i} \) such that for all \( c ∈ out(Q(t), σ_{A_i}) = \{c_1, \ldots, c_n\} \), \( t, c \models Π_i \). Such an \( A_i \)-move \( σ_{A_i} \) exists, since \( M, Q(t) \models q \) and \( q_i \subseteq q \). For each \( q_i \), in the considered semi-final tableau for \( θ \) that we are considering, for each \( c ∈ \{c_1, \ldots, c_n\} \) there exist at least an element of \( States(Π_i) \), say \( st_{i_j} \), such that \( M, Q(t.c_i) = st_{i_j} \). Such a state exists otherwise \( q \) would have been eliminated by rule \( E2 \). Thus for each \( q_i \) and for each \( c ∈ out(Q(t), σ_{A_i}) \) the couple \((c, st_{i_j})\) is an element of \( Ω \).

Let us observe that the construction of \((T_M, V, Q)\) may produce joker nodes in \( T_M \). So, for a given \( σ_{A_i} \in Δ_{A_i}(Q(t)) \), there exists a (possibly empty) set of all joker nodes \( t.c' \) such that \( c' ∈ out(σ_{A_i}, k) \). Note that in this case \( σ_{A_i} \) is both in \( Δ_{A_1}(Q(t)) \) and in \( Δ_{A_k} \), then \( V(t.c')(A) = Y \) and \( Q(t.c') = \emptyset \). Thus for all \( q' ∈ Q_{σ'_{A_i}}, f((||\{A\}||, V(t.c')) = true \). We can conclude that for each \((c, st_{i_j}) ∈ Ω\), either \( Q(t.c) \neq \emptyset \) and \( M, Q(t.c) = q \) or \( M, Q(t.c) = \emptyset \) and \( f((||\{A\}||, V(t)) = true \).

Now, let us reason on the treatment of the \( q'_i \), containing a negative next-time formula.

For each \( q'_i \), we can find in the pretableau of \( θ \), a prestate, say \( Π'_i \), equal to \( \{¬ψ_1, ψ_0, \ldots, ψ_{i-1}\} \). This prestate is the successor of \( q \) with a move vector \( σ \) corresponding to the case where all agents choose a negative formula and \( neg(σ) = i \). Then for each \( A_i \)-move \( σ_{A_i} \in Δ_{A_i} \), we choose a \( c_j ∈ out(Q(t), σ_{A_i}) = \{c_1, \ldots, c_n\} \), \( t, c_j \models Π'_i \). Such \( c_j \) exists, since \( M, Q(t) = q \) and \( q'_i \subseteq q \). For each \( c_j \), thus, in the considered semi-final tableau for \( θ \) that we are considering, for each \( c_j \) there exist at least an element of \( States(Π'_i) \), say \( st_{j} \), such that \( M, Q(t.c_j) = st_{j} \). Such states exist otherwise \( q \) would have been eliminated by
rule **E2.** Thus for each $q_i'$ and for each corresponding $c_j$ the couple $(c_j, st_j)$ is an element of $\Omega$.

Again, the construction of $\langle T_M, V, Q \rangle$ may produce joker nodes in $T_M$. So, for each $\sigma_{A,i} \in \Delta_A$ and $\not\in D_{A,i}(Q(t))$, there exists a (possibly empty) set whose elements are all the joker nodes $t.c'$ such that $c' \in out(\sigma_{A,i}, k)$. In this case $V(t.c')(A) = N$ and $Q(t.c') = \emptyset$. Thus for all $q_i' \in Q_{\sigma^*}$, $\mu((\langle A \rangle), V(t.c')) = \text{true}$. For each pair $(c, q) \in \Omega_{\langle A \rangle \bigcirc \varphi'}$, either $Q(t.c) \neq \emptyset$ and $M, Q(t.c) \models q$ or $Q(t.c) = \emptyset$ and $\mu((\langle A \rangle), V(t)) = \text{true}$.

At the end of the construction, $\Omega$ is a required set that satisfies

$$\rho(q, V(t)) = \bigwedge_{\varphi \in \text{PBHT}(q)} \mu(\varphi, V(t))$$

Successors of $y$ in $\langle T_r, r \rangle$ are all $y.i$ with $0 \leq i \leq |\Omega|$ and $r(y,i) = (c_i, q_i) \in \Omega$.

According to the case, several $A$-moves $\sigma_{A,i} \in \Delta_A$ and several vectors $c_i \in \Delta_{A_i}$ may be candidates for construction of $\Omega$ when $\varphi = \langle(A)\bigcirc \varphi' \rangle$ and $\varphi = \neg\langle(A)\bigcirc \varphi' \rangle$, respectively. For each possibility, a different set $\Omega$ may be obtained and therefore produces a different run $\langle T_r, r \rangle$. Let $\Omega$ be the set of all possible runs $\langle T_r, r \rangle$ constructed as above. The reasoning showing that $\Omega$ must contain at least an accepting run is identical to the one done in the soundness proof for the previous automaton, not using jokers. □

By using first Lemma 2 then Lemma 1 to make the obtained run a suitable one, we get the completeness result for the joker tree.

### 6 Concluding Remarks

For the well known temporal logic LTL, the relation holding between tableaux (as defined by [10]) and Büchi automata is well known. In this work, we have addressed a similar kind of questions for the case of ATL, showing that the main steps of the tableau procedure for a formula $\theta$ defined in [6] provide also the base for the construction of a particular kind of alternating automaton on infinite trees, named “joker automaton”. “joker automaton” can be used both to decide (tight) satisfiability and model check any CGS with respect to the property expressed by $\theta$, differently from the automaton proposed by [7]. In a next future, we plan to consider extensions/variations of ATL (or of related logics for multi-agent systems) allowing one to express also “irrevocable strategies of agents” (as defined, for instance in [2]) and to investigate, first, if suitable terminating tableaux systems can be defined and, secondly, their relation to automata-based decision procedure (if any).

**Acknowledgments.** The authors would like to thank M. Cialdea Mayer and V. Goranko for their useful remarks.
References


Appendix

Fig. 3. Complete pretableau $P^{\theta_1}$ for $\theta_1 = \neg\langle(1)\rangle\Box p \land \langle(1,2)\rangle\Box p \land \neg\langle(2)\rangle\Diamond \neg p$. 

Γ
θ
= ¬⟨⟨1⟩⟩□p ∧ ⟨⟨1,2⟩⟩□p ∧ ¬⟨⟨2⟩⟩□¬p

Γ
θ

Γ
θ

Γ
θ

Γ
θ

Γ
θ

Γ
θ

Γ
θ

Γ
θ

Γ
θ
Fig. 4. Semi-final and final tableau $T^\theta_0$ is open, hence the ATL-formula $\theta_1$ is satisfiable.

Fig. 5. Final tableau $T^{\theta_1}$ obtained with the new tableau procedure.

Fig. 6. Non accepted run of $\mathfrak{A}_{\theta_1}$ over the $\{p\}$-labelled joker tree $(T_{M_1}, V, \mathcal{Q})$. 