ATL with Strategy Contexts: Expressiveness and Model Checking

Arnaud Da Costa[1], François Laroussinie[2], and Nicolas Markey[1]

1   Lab. Spécification & Vérification, ENS Cachan & CNRS, France
2   LIAFA, Univ. Paris Diderot - Paris 7 & CNRS, France

─── **Abstract** ───

We study the alternating-time temporal logics ATL and ATL* extended with *strategy contexts*: these make agents *commit to their strategies* during the evaluation of formulas, contrary to plain ATL and ATL* where strategy quantifiers *reset* previously selected strategies.

We illustrate the important expressive power of strategy contexts by proving that they make the extended logics, namely ATL$_{sc}$ and ATL$^\star_{sc}$, equally expressive: any formula in ATL$^\star_{sc}$ can be translated into an equivalent, linear-size ATL$_{sc}$ formula. Despite the high expressiveness of these logics, we prove that their model-checking problems remain decidable by designing a tree-automata-based algorithm for model-checking ATL$^\star_{sc}$ on the full class of n-player concurrent game structures.

1   Introduction

Temporal logics and model checking.   Thirty years ago, temporal logics (LTL, CTL) have been proposed for specifying properties of reactive systems, with the aim of automatically checking that those properties hold for these systems [18, 10, 19]. This *model-checking* approach to formal verification has been widely studied, with powerful algorithms and implementations, and successfully applied in many situations.

Alternating-time temporal logic (ATL).   In the last ten years, temporal logics have been extended with the ability of specifying *controllability properties* of *multi-agent systems*: the evolution of a multi-agent system depends on the concurrent actions of several agents, and ATL extends CTL with *strategy quantifiers* [4]: it can express properties such as *agent A has a strategy to keep the system in a set of safe states, whatever the other agents do.*

Nesting strategy quantifiers.   Assume that, in the formula above, "safe states" are those from which agent B has a strategy to reach her goal state q_B infinitely often, and consider the system depicted on Fig. 1, where the circled states are controlled by player A (meaning that Player A selects the transition to be fired from those state) and the square state is controlled by player B. It is easily seen that this game contains no "safe state": after each visit to q_B, Player A can decide to take the system to the rightmost state, from which q_B is not reachable. It follows that Player A has no strategy to keep the system in safe states.

Figure 1 Example of a two-player turn-based game

Now, assume that Player A commits to always select the transition to the left, when the system is in the initial (double-circled) state. Then under this strategy, it suffices for
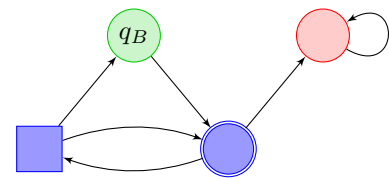
Player B to always go to q_B when the system is in the square state in order to achieve her goal of visiting q_B infinitely often. The difference with the previous case is that here, Player B *takes advantage* of Player A's strategy in order to achieve her goal.

Both interpretations of our original property can make sense, depending on the context. However, the original semantics of ATL cannot capture the second interpretation: strategy quantifications in ATL "reset" previous strategies. While this is very convenient algorithmically (and makes ATL model-checking polynomial-time for some game models), it prevents ATL from expressing many interesting properties of games (especially non-zero-sum games).

In [7], we introduced an alternative semantics for ATL, where strategy quantifiers *store* strategies in a *context*. Those strategies then apply for evaluating the whole subformula, until they are explicitly removed from the context or replaced with a new strategy. We demonstrated the high expressiveness of this new semantics by showing that it can express important requirements, *e.g.* existence of equilibria or dominating strategies.

Our contribution. This work is a continuation of [7]. Our contribution in this paper is twofold: on the one hand, we prove that $\mathsf{ATL}^\star_{sc}$ is not more expressive than ATL_{sc}: this is a theoretical argument witnessing the expressive power of strategy contexts; it complements the more practical arguments presented in [7]. On the other hand, we develop an algorithm for $\mathsf{ATL}^\star_{sc}$ model-checking, based on alternating tree automata. Our algorithm uses a novel encoding of strategies into the execution tree of the underlying concurrent game structures. This way, it is valid for the whole class of concurrent game structures and without restrictions on strategies, contrary to previously existing algorithms on related extensions of ATL.

Related work. In the last three years, several approaches have been proposed to increase the expressiveness of ATL and $\mathsf{ATL}^\star$.

- *Strategy logic* [8, 9] extends LTL with first-order quantification over strategies. This allows for very expressive constructs: for instance, the property above would be written as $\exists \sigma_A. \; [\mathbf{G} \; (\exists \sigma_B. \; (\mathbf{G}\,\mathbf{F} \; q_B)\,(\sigma_A, \sigma_B))]\,(\sigma_A)$.

  This logic was only studied on two-player turn-based games in [8, 9], where a non-elementary algorithm is given. The algorithm we propose in this paper could be adapted to handle strategy logic in multi-player concurrent games.

- $\mathsf{Q}D\mu$ [17] is a second-order extension of the propositional μ-calculus augmented with decision modalities. In terms of expressiveness, fixpoints allow for richer constructs than CTL- or LTL-based approaches. Again, model-checking has been proved to be decidable, but only over the class of alternating transition systems (as defined in [3]).

- *Stochastic game logic* [6] is an extension of ATL similar to ours, but in the stochastic case. It is proved undecidable in the general case, and decidable when strategy quantification is restricted to memoryless (randomized or deterministic) strategies.

- several other semantics of ATL, related to ours, are discussed in [1, 2]. A Δ^P_2-algorithm is proposed there for a subclass of our logic (where strategies stored in the context are *irrevocable* and cannot be overwritten), but no proof of correctness is given. In [20], an NP algorithm is proposed for the same subclass, but where strategy quantification is restricted to memoryless strategies.

By lack of space, some proofs are omitted in this paper, but they are detailed in [11].

2   ATL with strategy contexts

2.1   Concurrent game structures.

Concurrent game structures [4] are a multi-player extension of classical Kripke structures. Their definition is as follows:

▶ **Definition 1.** A *Concurrent Game Structure* (*CGS* for short) $\mathcal{C}$ is an 7-tuple $\langle \mathsf{Loc}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ where:

- $\langle \mathsf{Loc}, \mathsf{Lab}, \delta \rangle$ is a finite Kripke structure, where Loc is the set of *locations*, $\mathsf{Lab}\colon \mathsf{Loc} \to 2^{\mathsf{AP}}$ is a labelling function, and $\delta \subseteq \mathsf{Loc} \times \mathsf{Loc}$ is the set of transitions;
- $\mathsf{Agt} = \{A_1, ..., A_p\}$ is a finite set of *agents* (or *players*);
- $\mathcal{M}$ is a finite, non-empty set of moves;
- $\mathsf{Mov}\colon \mathsf{Loc} \times \mathsf{Agt} \to \mathcal{P}(\mathcal{M}) \smallsetminus \{\varnothing\}$ defines the (finite) set of possible moves of each agent in each location.
- $\mathsf{Edg}\colon \mathsf{Loc} \times \mathcal{M}^{\mathsf{Agt}} \to \delta$ is a transition table; with each location ℓ and each set of moves of the agents, it associates the resulting transition, which is required to depart from ℓ.

The size $|\mathcal{C}|$ of a CGS $\mathcal{C}$ is $|\mathsf{Loc}| + |\mathsf{Edg}|$, where $|\mathsf{Edg}|$ is the size of the transition table[1].

The intended behaviour of a CGS is as follows [4]: in a location ℓ, each player A_i in Agt chooses one among her possible moves m_i in $\mathsf{Mov}(\ell, A_i)$; the next transition to be fired is given by $\mathsf{Edg}(\ell, (m_1, ..., m_p))$. We write $\mathsf{Next}(\ell)$ for the set of all transitions corresponding to possible moves from ℓ, and $\mathsf{Next}(\ell, A_j, m_j)$, with $m_j \in \mathsf{Mov}(\ell, A_j)$, for the restriction of $\mathsf{Next}(\ell)$ to possible transitions from ℓ when player A_j plays the move m_j. We extend Mov and Next to coalitions (*i.e.*, sets of agents) in the natural way:

- given $A \subseteq \mathsf{Agt}$ and $\ell \in \mathsf{Loc}$, $\mathsf{Mov}(\ell, A)$ denotes the set of possible moves for coalition A from ℓ. Those moves m are composed of one single move per agent of the coalition, *i.e.*, $m = (m_a)_{a \in A}$.
- Given $m = (m_a)_{a \in A} \in \mathsf{Mov}(\ell, A)$, we let $\mathsf{Next}(\ell, A, m)$ denote the restriction of $\mathsf{Next}(\ell)$ to locations reachable from ℓ when every player $A_j \in A$ makes the move m_{A_j}.

A (finite or infinite) *path* of $\mathcal{C}$ is a sequence $\rho = \ell_0 \ell_1 \ldots$ of locations such that for any i, $\ell_{i+1} \in \mathsf{Next}(\ell_i)$. Finite paths are also called *history*. The length of a history $\rho = \ell_0 \ell_1 \ldots \ell_n$ is n. We write $\rho^{i \to j}$ for the part of ρ between ℓ_i and ℓ_j (inclusive). In particular, $\rho^{i \to j}$ is empty iff $j < i$. We simply write ρ^i for $\rho^{i \to i}$, denoting the $i+1$-st location ℓ_i of ρ. We also define $\mathsf{first}(\rho) = \rho^0$, and, if ρ has finite length n, $\mathsf{last}(\rho) = \rho^n$. Given a history π of length n and a path ρ s.t. $\mathsf{last}(\pi) = \mathsf{first}(\rho)$, the concatenation of π and ρ is the path $\tau = \pi \cdot \rho$ s.t. $\tau^{0 \to n} = \pi$ and $\tau^{n \to \infty} = \rho$ (notice that the last location of π and the first location of ρ are "merged").

A *strategy* for a player $A_i \in \mathsf{Agt}$ is a function f_i that maps any history to a possible move for A_i, *i.e.*, satisfying $f_i(\ell_0 \ldots \ell_m) \in \mathsf{Mov}(\ell_m, A_i)$. A strategy for a coalition A of agents is a mapping assigning a strategy to each agent in the coalition. The set of strategies for A is denoted $\mathsf{Strat}(A)$. The *domain* of $F_A \in \mathsf{Strat}(A)$ (denoted $\mathsf{dom}(F_A)$) is A. Given a coalition B, the strategy $(F_A)_{|B}$ (resp. $(F_A)_{\smallsetminus B}$) denotes the restriction of F_A to coalition $A \cap B$ (resp. $A \smallsetminus B$).

Let ρ be a history of length n. A strategy $F_A = (f_j)_{A_j \in A}$ for some coalition A induces a set of paths from ρ, called the *outcomes* of F_A after (or from) ρ, and denoted $\mathsf{Out}(\rho, F_A)$:

---

[1] Our results would still hold (with the same complexity) if we consider symbolic CGSs [13], where the transition table is encoded succinctly as boolean formulas.

a path $\pi = \rho \cdot \ell_1 \ell_2 \ldots$ is in $\mathsf{Out}(\rho, F_A)$ iff, writing $\ell_0 = \mathsf{last}(\rho)$, for all $i \geq 0$ there exists a set of moves $(m_k^i)_{A_k \in \mathsf{Agt}}$ such that $m_k^i \in \mathsf{Mov}(\ell_i, A_k)$ for all $A_k \in \mathsf{Agt}$, $m_k^i = f_{A_k}(\pi^{0 \to n+i})$ if $A_k \in A$, and $\ell_{i+1} \in \mathsf{Next}(\ell_i, \mathsf{Agt}, (m_k^i)_{A_k \in \mathsf{Agt}})$. We write $\mathsf{Out}^\infty(\rho, F_A)$ for the set of infinite outcomes of F_A after ρ. Note that $\mathsf{Out}(\rho, F_A) \subseteq \mathsf{Out}(\rho, (F_A)_{|B})$ for any two coalitions A and B, and that $\mathsf{Out}(\rho, F_\varnothing)$ represents the set of all paths starting with ρ.

It is also possible to *combine* two strategies $F \in \mathsf{Strat}(A)$ and $F' \in \mathsf{Strat}(B)$, resulting in a strategy $F \circ F' \in \mathsf{Strat}(A \cup B)$ defined as follows: $(F \circ F')_{|A_j}(\rho)$ is $F_{|A_j}(\rho)$ (resp. $F'_{|A_j}(\rho)$) if $A_j \in A$ (resp. $A_j \in B \smallsetminus A$).

Finally, given a strategy F and a history ρ, we define the strategy F^ρ corresponding to the behaviour of F after prefix ρ: it is defined, for any history π with $\mathsf{last}(\rho) = \mathsf{first}(\pi)$, as $F^\rho(\pi) = F(\rho \cdot \pi)$.

2.2 Alternating-time temporal logics.

The logics ATL and $\mathsf{ATL}^\star$ have been defined in [4] as extensions of CTL and $\mathsf{CTL}^\star$ with strategy quantification. Following [7], we further extend them with *strategy contexts*:

▶ **Definition 2.** The syntax of $\mathsf{ATL}^\star_{sc}$ is defined by the following grammar:

$$\mathsf{ATL}^\star_{sc} \ni \varphi_s, \psi_s ::= p \mid \neg\varphi_s \mid \varphi_s \vee \psi_s \mid \langle \cdot A \rangle \varphi_p \mid \rangle A \langle \varphi_s$$
$$\varphi_p, \psi_p ::= \varphi_s \mid \neg\varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X}\,\varphi_p \mid \varphi_p\,\mathbf{U}\,\psi_p$$

with $p \in \mathsf{AP}$ and $A \subseteq \mathsf{Agt}$. Formulas defined as φ_s are called *state-formulas*, while φ_p defines *path-formulas*. The logic ATL_{sc} is obtained by restricting the grammar of $\mathsf{ATL}^\star_{sc}$ path-formulas as follows:

$$\varphi_p, \psi_p ::= \neg\varphi_p \mid \mathbf{X}\,\varphi_s \mid \varphi_s\,\mathbf{U}\,\psi_s.$$

That a formula φ in $\mathsf{ATL}^\star_{sc}$ (or ATL_{sc}) holds (initially) along a computation ρ of a CGS $\mathcal{C}$ under a strategy context F (*i.e.*, a preselected strategy for some of the players, hence belonging to some $\mathsf{Strat}(A)$ for a coalition A), denoted $\mathcal{C}, \rho \models_F \varphi$, is defined as follows:

$$
\begin{aligned}
\mathcal{C}, \rho &\models_F p &&\text{iff}\quad p \in \mathsf{Lab}(\mathsf{first}(\rho)) \\
\mathcal{C}, \rho &\models_F \neg\varphi &&\text{iff}\quad \mathcal{C}, \rho \not\models_F \varphi \\
\mathcal{C}, \rho &\models_F \varphi \vee \psi &&\text{iff}\quad \mathcal{C}, \rho \models_F \varphi \text{ or } \mathcal{C}, \rho \models_F \psi \\
\mathcal{C}, \rho &\models_F \langle \cdot A \rangle \varphi_p &&\text{iff}\quad \exists F_A \in \mathsf{Strat}(A).\quad \forall \rho' \in \mathsf{Out}^\infty(\mathsf{first}(\rho), F_A \circ F).\ \mathcal{C}, \rho' \models_{F_A \circ F} \varphi_p \\
\mathcal{C}, \rho &\models_F \rangle A \langle \varphi_s &&\text{iff}\quad \mathcal{C}, \rho \models_{F \smallsetminus A} \varphi_s \\
\mathcal{C}, \rho &\models_F \mathbf{X}\,\varphi_p &&\text{iff}\quad \mathcal{C}, \rho^{1 \to \infty} \models_{F^{\rho^{0 \to 1}}} \varphi_p \\
\mathcal{C}, \rho &\models_F \varphi_p\,\mathbf{U}\,\psi_p &&\text{iff}\quad \exists i \geq 0.\ \mathcal{C}, \rho^{i \to \infty} \models_{F^{\rho^{0 \to i}}} \psi_p \text{ and } \forall 0 \leq j < i.\ \mathcal{C}, \rho^{j \to \infty} \models_{F^{\rho^{0 \to j}}} \varphi_p
\end{aligned}
$$

We define the following shorthands, which will be useful in the sequel: $\top \stackrel{\text{def}}{=} p \vee \neg p$, $\bot \stackrel{\text{def}}{=} \neg\top$, $\mathbf{F}\,\varphi \stackrel{\text{def}}{=} \top\,\mathbf{U}\,\varphi$, $\mathbf{G}\,\varphi \stackrel{\text{def}}{=} \neg\mathbf{F}\,\neg\varphi$, $\langle \cdot A \rangle \varphi_s \stackrel{\text{def}}{=} \langle \cdot A \rangle (\bot\,\mathbf{U}\,\varphi_s)$ and $\langle\!\langle A \rangle\!\rangle \varphi \stackrel{\text{def}}{=} \rangle\mathsf{Agt}\langle\ \langle \cdot A \rangle \varphi$.

▶ **Example 3** (see [7] for more examples). We illustrate the usefulness of strategy contexts with some examples. First, the last shorthand $\langle\!\langle A \rangle\!\rangle$ is the classical $\mathsf{ATL}^\star$ strategy quantifier (where each quantification resets the context), so that ATL_{sc} and $\mathsf{ATL}^\star_{sc}$ encompass ATL and $\mathsf{ATL}^\star$, respectively.

ATL_{sc} can also express qualitative equilibria properties, for instance Nash equilibria. Given the (non-zero-sum) objectives Φ_1 and Φ_2 of players 1 and 2, Nash equilibria are strategy profiles where none of the player can unilaterally improve her payoff. In other

terms, if the Player-1 strategy in the context is not winning against the Player-2 strategy, then there is no Player-1 winning strategy against this particular strategy of Player 2 (and symmetrically). Thus, the existence of a Nash equilibrium can be expressed as

$$\langle A_1, A_2 \rangle \left[\left(\langle A_1 \rangle \, \Phi_1 \to \Phi_1 \right) \wedge \left(\langle A_2 \rangle \, \Phi_2 \to \Phi_2 \right) \right]$$

As another example, we mention the interaction between a server S and different clients $(C_i)_i$, where we may want to express that the server can be programmed in such a way that each client C_i has a strategy to have its request granted. This could be written as

$$\langle \cdot S \rangle \, \mathbf{G} \left[\bigwedge_i \left(\mathsf{req}_i \to \langle A_i \rangle \, \mathbf{F} \, \mathsf{grant}_i \right) \right]$$

As stated in Lemma 4, the truth value of a *state* formula φ_s depends only on the strategy context F and the *first* state of the computation ρ where it is interpreted (thus we may simply write $\mathcal{C}, \mathsf{first}(\rho) \models_F \varphi_s$ when it raises no ambiguity).

▶ **Lemma 4.** *Let $\mathcal{C}$ be a CGS, and $F \in \mathsf{Strat}(A)$ be a strategy context. For any state formula φ_s, and for any two infinite paths ρ and ρ' with $\mathsf{first}(\rho) = \mathsf{first}(\rho')$, it holds*

$$\mathcal{C}, \rho \models_F \varphi_s \qquad \Leftrightarrow \qquad \mathcal{C}, \rho' \models_F \varphi_s.$$

Proof. The proof is by induction on the structure of φ_s: the result obviously holds for atomic propositions, and it is clearly preserved by boolean combinations and by the $\cdot \rangle A \langle$ operator. Finally, if $\varphi_s = \langle A \rangle \, \psi_s$, the result is immediate as the semantics only involves the first location of the path along which the formula is being evaluated.   ◀

▶ Remark. Note that contrary to ATL, it is not possible to restrict to memoryless strategies (*i.e.*, that only depend on the current state) for ATL_{sc} formulas. For example, the formula $\langle A \rangle \, \mathbf{G} \, (\langle \varnothing \rangle \, \mathbf{F} \, P \wedge \langle \varnothing \rangle \, \mathbf{F} \, P')$ is equivalent in a standard Kripke struture (seen as a CGS with one single player A) to the $\mathsf{CTL}^\star$ formula $\mathsf{E}(\overset{\infty}{\mathbf{F}} P \wedge \overset{\infty}{\mathbf{F}} P')$ that may require strategies with memory. The next section provides more results on the extra expressiveness brought in by strategy contexts.

3   The expressive power of strategy contexts

As shown in [7], adding strategy contexts in formulas increases the expressive power of logics: ATL_{sc} (resp. $\mathsf{ATL}_{sc}^\star$) is strictly more expressive than ATL (resp. $\mathsf{ATL}^\star$). Game Logic (see [4]) can also be translated into $\mathsf{ATL}_{sc}^\star$ (while the converse is not true). In this section, we present some new results on the expressiveness of ATL_{sc}.

3.1   Alternating bisimulation.

Contrary to ATL, $\mathsf{ATL}^\star$, GL or AMC, our logics are not alternating-bisimulation invariant (see [5]), indeed we have:

▶ **Lemma 5.** *There exists two CGSs $\mathcal{C}$ and $\mathcal{C}'$, with an alternating-bisimulation linking two states ℓ_0 of $\mathcal{C}$ and ℓ_0' in $\mathcal{C}'$, and an ATL_{sc} formula φ such that $\mathcal{C}, \ell_0 \models \varphi$ and $\mathcal{C}', \ell_0' \not\models \varphi$.*

3.2 Relative expressiveness of ATL_{sc} and $\mathsf{ATL}_{sc}^\star$.

Surprisingly, strategy contexts bring ATL_{sc} to the same expressiveness as $\mathsf{ATL}_{sc}^\star$. This was already exemplified above, with the $\mathsf{CTL}^\star$ formula $\mathsf{E}(\overset{\approx}{\mathbf{F}} P \wedge \overset{\approx}{\mathbf{F}} P')$. We can extend this approach to any $\mathsf{ATL}_{sc}^\star$ formula: the idea is to

1. first use *full* strategy contexts (by adding universally quantified strategies) in order to be able to insert the $\langle\cdot\varnothing\cdot\rangle$ modality before every temporal modality, and
2. ensure that for every nested strategy quantifier $\langle\cdot A\cdot\rangle$, Coalition A cannot take advantage of the added strategies.

Given an $\mathsf{ATL}_{sc}^\star$ formula Φ and a coalition B, we define $\widehat{\Phi}^{[B]}$ inductively as follows:

$$\widehat{P}^{[B]} \overset{\text{def}}{=} P \qquad \widehat{\neg\varphi}^{[B]} \overset{\text{def}}{=} \neg\widehat{\varphi}^{[B]} \qquad \widehat{\varphi \wedge \psi}^{[B]} \overset{\text{def}}{=} \widehat{\varphi}^{[B]} \wedge \widehat{\psi}^{[B]}$$

$$\widehat{\mathbf{X}\,\varphi}^{[B]} \overset{\text{def}}{=} \langle\cdot\varnothing\cdot\rangle\,\mathbf{X}\,\widehat{\varphi}^{[B]} \qquad\qquad \widehat{\varphi\,\mathbf{U}\,\psi}^{[B]} \overset{\text{def}}{=} \langle\cdot\varnothing\cdot\rangle\,(\widehat{\varphi}^{[B]}\,\mathbf{U}\,\widehat{\psi}^{[B]})$$

$$\widehat{\langle\cdot A\cdot\rangle\,\varphi}^{[B]} \overset{\text{def}}{=} \langle\cdot A\cdot\rangle\,\neg\,\langle\cdot\mathsf{Agt}\backslash(A\cup B)\cdot\rangle\,\neg\widehat{\varphi}^{[A\cup B]} \qquad \widehat{\,\rangle A\langle\cdot\,\varphi}^{[B]} \overset{\text{def}}{=} \widehat{\varphi}^{[B\smallsetminus A]}$$

Clearly, $\widehat{\Phi}^{[B]}$ is an ATL_{sc} formula. The idea behind this translation is that a state-formula $\widehat{\varphi}^A$ interpreted in a strategy context F only depends on $F_{|A}$. We then have:

▶ **Lemma 6.** *Let $\mathcal{C}$ be a CGS, ℓ be one of its locations, and F be a strategy context. Then for any $\mathsf{ATL}_{sc}^\star$ formula φ, for any strategy context G s.t. $\mathsf{dom}(G) = \mathsf{Agt} \smallsetminus \mathsf{dom}(F)$, and for any outcome $\pi \in \mathsf{Out}^\infty(\ell, G \circ F)$, it holds: $\mathcal{C}, \pi \models_F \varphi \Leftrightarrow \mathcal{C}, \pi \models_{G \circ F} \widehat{\varphi}^{[\mathsf{dom}(F)]}$. Moreover, if φ is a state-formula, this result extends to any strategy context G s.t. $\mathsf{dom}(G) \cap \mathsf{dom}(F) = \varnothing$.*

Since our transformation does not depend on the underlying CGS, we get:

▶ **Theorem 7.** *Given a set of agents Agt, any $\mathsf{ATL}_{sc}^\star$ formula φ can be translated into an equivalent (under the empty context) ATL_{sc} formula $\widehat{\varphi}$ for any CGS based on Agt.*

Another consequence of the previous result is that any $\mathsf{ATL}^\star$ state-formula φ can be translated into the equivalent ATL_{sc} formula $\widehat{\varphi}^\varnothing$ in polynomial time. Thus we have:

▶ **Corollary 8.** *Model-checking ATL_{sc} is $2\mathsf{EXPTIME}$-hard.*

4 From ATL_{sc} to alternating tree automata

The main result of this section is the following:

▶ **Theorem 9.** *Model-checking ATL_{sc} formulas with at most k nested strategy quantifiers can be achieved in $(k+1)\mathsf{EXPTIME}$. The program complexity (i.e., the complexity of model-checking a fixed ATL_{sc} formula) is in $\mathsf{EXPTIME}$.*

The proof mainly consists in building an alternating tree automaton from a formula and a CGS. Similar approaches have already been proposed for strategy logic [9] or $\mathrm{Q}D\mu$ [17], but they were only valid for subclasses of CGSs: strategy logic was only studied on turn-based games, while the algorithm for $\mathrm{Q}D\mu$ was restricted to ATSs [3]. In both cases, the important point is that strategies are directly encoded as trees, with as many successors of a node as the number of possible moves from the corresponding node. With this representation, it is required that two different successors of a node correspond to two different states (which is the case for ATSs, hence for turn-based games): if this is not the case, the tree automaton may accept strategies that do not only depend on the sequence of states visited in the history,

but also on the sequence of moves proposed by the players. Our encoding is different: we work on the execution tree of the CGS under study, and label each node with possible moves of the players. We then have to focus on branches that correspond to outcomes of selected strategies, and check that they satisfy the requirement specified by the formula. Before presenting the detailed proof, we first introduce alternating tree automata and fix notations.

4.1   Trees and alternating tree automata

Let Σ and S be two finite sets. A Σ-*labelled S-tree* is a pair $\mathcal{T} = \langle T, l \rangle$, where

- $T \subseteq S^*$ is a non-empty set of finite words on S satisfying the following constraints: for any non-empty word $n = m \cdot s$ in T with $m \in T$ and $s \in S$, the word m is also in T;
- $l: T \to \Sigma$ is a labeling function.

Given such a tree $\mathcal{T} = \langle T, l \rangle$ and a node $n \in T$, the set of *directions from n in T* is the set $\mathsf{dir}_T(n) = \{s \in S \mid n \cdot s \in T\}$. The set of *successors of n in T* is $\mathsf{succ}_T(n) = \{n \cdot s \mid s \in \mathsf{dir}_T(n)\}$. We use $\mathcal{T}_n$ to denote the subtree rooted in n. An S-tree is complete if $T = S^*$, *i.e.*, if $\mathsf{dir}_T(n) = S$ for all $n \in T$. We may omit the subscript T when it is clear from the context.

The set of *infinite paths of* $\mathcal{T}$ is the set $\mathsf{Path}_\mathcal{T} = \{s_0 \cdot s_1 \cdots \in S^\omega \mid \forall i \in \mathbb{N}.\ s_0 \cdot s_1 \cdots s_i \in T\}$. Given such an infinite path $\pi = (s_i)_{i \in \mathbb{N}}$, we write $l(\pi)$ for the infinite sequence $(l(s_i))_{i \in \mathbb{N}} \in \Sigma^\omega$, and $\mathsf{Inf}(l(\pi))$ for the set of letters in Σ that appear infintely often along $l(\pi)$.

Assume that $\Sigma = \Sigma_1 \times \Sigma_2$, and pick a Σ-labelled S-tree $\mathcal{T} = \langle T, l \rangle$. For all $n \in T$, we write $l(n) = (l_1(n), l_2(n))$ with $l_i(n) \in \Sigma_i$ for $i \in \{1, 2\}$. Then for $i \in \{1, 2\}$, the *projection of* $\mathcal{T}$ *on* Σ_i, denoted by $\mathsf{proj}_{\Sigma_i}(\mathcal{T})$, is the Σ_i-labelled S-tree $\langle T, l_i \rangle$. Two Σ-labelled S-trees are Σ_i-*equivalent* if their projections on Σ_i are equal. These notions naturally extend to more complex alphabets, of the form $\prod_{i \in I} \Sigma_i$.

We now define alternating tree automata, which will be used in the proof. This requires the following definition: the set of *positive boolean formulas* over a finite set P of propositional variables is the set of formulas generated by: $\mathsf{PBF}(P) \ni \zeta ::= p \mid \zeta \wedge \zeta \mid \zeta \vee \zeta \mid \top \mid \bot$ where p ranges over P. That a valuation $v: P \to \{\top, \bot\}$ satisfies a formula in $\mathsf{PBF}(P)$ is defined in the natural way. We abusively say that a subset P' of P satisfies a formula $\varphi \in \mathsf{PBF}(P)$ iff the valuation $\mathbb{1}_{P'}$ (mapping the elements of P' to $\top$ and te elements of $P \smallsetminus P'$ to $\bot$) satisfies φ. Since negation is not allowed, if $P' \models \varphi$ and $P' \subseteq P''$, then also $P'' \models \varphi$.

▶ **Definition 10.** Let S and Σ be two finite sets. An *alternating S-tree automaton on* Σ, or $\langle S, \Sigma \rangle$-ATA, is a 4-tuple $\mathcal{A} = \langle Q, q_0, \tau, \mathsf{Acc} \rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, Σ is a finite alphabet, $\tau: Q \times \Sigma \to \mathsf{PBF}(S \times Q)$ is the transition function, and $\mathsf{Acc}: Q^\omega \to \{\top, \bot\}$ is the acceptance function.

A *non-deterministic S-tree automaton on* Σ, or $\langle S, \Sigma \rangle$-NTA, is a $\langle S, \Sigma \rangle$-ATA in which conjunctions are not allowed for defining the transition function. The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the number of states in Q.

Let $\mathcal{A} = \langle Q, q_0, \tau, \mathsf{Acc} \rangle$ be an $\langle S, \Sigma \rangle$-ATA, and $\mathcal{T} = \langle T, l \rangle$ be a Σ-labelled S-tree. An *execution tree* of $\mathcal{A}$ on $\mathcal{T}$ is a $T \times Q$-labelled $S \times Q$-tree $\mathcal{E} = \langle E, p \rangle$ such that $p(\varepsilon) = (\varepsilon, q_0)$, and for each node $e \in E$ with $p(e) = (t, q)$, the set $\mathsf{dir}_E(e) = \{(s_0, q_0), (s_1, q_1), ..., (s_n, q_n)\} \subseteq S \times Q$ satisfies $\tau(q, l(t))$, and for all $0 \leq i \leq n$, the node $e \cdot (s_i, q_i)$ is labelled with $(t \cdot s_i, q_i)$. We write $p_S(e \cdot (s_i, q_i)) = t \cdot s_i$ and $p_Q(e \cdot (s_i, q_i)) = q_i$ for the two components of the labelling function.

An execution tree is *accepting* if $\mathsf{Acc}(p_Q(\pi)) = \top$ for any infinite path $\pi \in (S \times Q)^\omega$ in $\mathsf{Path}_\mathcal{E}$. A tree $\mathcal{T}$ is accepted by $\mathcal{A}$ iff there exists an accepting execution tree of $\mathcal{A}$ on $\mathcal{T}$. In the sequel, we use *parity acceptance condition*, given as a function $\Omega: Q \to \{0, ..., k-1\}$, from which Acc is defined as follows: $\mathsf{Acc}(p_Q(\pi)) = \top$ iff $\min\{\Omega(q) \mid q \in \mathsf{Inf}(p_Q(\pi))\}$ is

even. $\langle S, \Sigma \rangle$-ATAs with such accepting conditions are called $\langle S, \Sigma \rangle$-APTs, and given an $\langle S, \Sigma \rangle$-APT $\mathcal{A}$, the size of the image of Ω is called the *index* of $\mathcal{A}$, and is denoted by $\mathsf{idx}(\mathcal{A})$. Analogously, $\langle S, \Sigma \rangle$-NPTs are $\langle S, \Sigma \rangle$-NTAs with parity acceptance conditions.

4.2 Unwinding of a CGS

Let $\mathcal{C} = \langle \mathsf{Loc}, \mathsf{Lab}, \delta, \mathsf{Agt}, \mathcal{M}, \mathsf{Mov}, \mathsf{Edg} \rangle$ be an n-player CGS, where we assume w.l.o.g. that $\delta = \mathsf{Loc} \times \mathsf{Loc}$, and $\mathsf{Mov}(\ell, A_i) = \mathcal{M}$ for any state ℓ and any player A_i. Let ℓ_0 be a state of $\mathcal{C}$.

For each location $\ell \in \mathsf{Loc}$, we define $\Sigma(\ell) = \{\ell\} \times \{\mathsf{Lab}(\ell)\} \times \{\mathsf{Edg}(\ell)\}$, and $\Sigma^+(\ell) = \Sigma(\ell) \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt}} \times 2^{\{p_o, p_l, p_r\}}$, where $\bot$ is a special symbol not in $\mathcal{M}$ and p_o, p_l and p_r are three fresh propositions not in AP. We let[2] $\Sigma_{\mathcal{C}} = \bigcup_{\ell \in \mathsf{Loc}} \Sigma(\ell)$, and $\Sigma_{\mathcal{C}}^+ = \bigcup_{\ell \in \mathsf{Loc}} \Sigma^+(\ell)$.

The *unwinding* of $\mathcal{C}$ from ℓ_0 is the $\Sigma_{\mathcal{C}}$-labelled complete Loc-tree $\mathcal{U} = \langle U, v \rangle$ where $U = \mathsf{Loc}^*$ and $v(u) \in \Sigma(\mathsf{last}(\ell_0 \cdot u))$ for all $u \in U$. An *extended unwinding* of $\mathcal{C}$ from ℓ_0 is a $\Sigma_{\mathcal{C}}^+$-labelled complete Loc-tree $\mathcal{U}'$ such that $\mathsf{proj}_{\Sigma_{\mathcal{C}}}(\mathcal{U}') = \mathcal{U}$. For each letter σ of $\Sigma_{\mathcal{C}}^+$, we write σ_{Loc}, σ_{AP}, σ_{Edg}, σ_{str} and σ_p for the five components, and extend this subscripting notation for the labelling functions of trees (written l_{Loc}, l_{AP}, l_{Edg}, l_{str} and l_p).

In the sequel, we identify a node u of $\mathcal{U}$ (which is a finite word over Loc) with the finite path $\ell_0 \cdot u$ of $\mathcal{C}$. Notice that this sequence of states of $\mathcal{C}$ may correspond to no *real* path of $\mathcal{C}$, in case it involves a transition that is not in the image of Edg.

With $\mathcal{C}$ and ℓ_0, we associate a $\langle \mathsf{Loc}, \Sigma_{\mathcal{C}}^+ \rangle$-APT $\mathcal{A}_{\mathcal{C}, \ell_0} = \langle \overline{\mathsf{Loc}}, \overline{\ell_0}, \tau, \Omega \rangle$ s.t. $\overline{\mathsf{Loc}} = \{\overline{\ell} \mid \ell \in \mathsf{Loc}\}$, $\overline{\ell_0}$ is the initial state, Ω constantly equals 0 (hence any valid execution tree is accepting), and given a state $\overline{\ell} \in \overline{\mathsf{Loc}}$ and a letter $\sigma \in \Sigma_{\mathcal{C}}^+$, the transition function is defined as follows: if $\sigma \in \Sigma^+(\ell)$, we let $\tau(\overline{\ell}, \sigma) = \bigwedge_{\ell' \in \mathsf{Loc}} (\ell', \overline{\ell'})$, and otherwise, we let $\tau(\overline{\ell}, \sigma) = \bot$.

▶ **Lemma 11.** *Let $\mathcal{C}$ be a CGS and ℓ_0 be a state of $\mathcal{C}$. Let $\mathcal{T} = \langle T, l \rangle$ be a $\Sigma_{\mathcal{C}}^+$-labelled Loc-tree. Then $\mathcal{A}_{\mathcal{C}, \ell_0}$ accepts $\mathcal{T}$ iff $\mathsf{proj}_{\Sigma_{\mathcal{C}}}(\mathcal{T})$ is the unwinding of $\mathcal{C}$ from ℓ_0.*

In the sequel, we also use automaton $\mathcal{A}_{\mathcal{C}}$, which accepts the union of all $\mathcal{L}(\mathcal{A}_{\mathcal{C}, \ell_0})$ when ℓ_0 ranges over Loc. It is easy to come up with such an automaton, *e.g.* with Lemma 14 below.

4.3 Strategy quantification

Let $\mathcal{T} = \langle T, l \rangle$ be a $\Sigma_{\mathcal{C}}^+$-labelled complete Loc-tree accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$. Such a tree defines *partial* strategies for each player: for $A \in \mathsf{Agt}$, and for each node $n \in T$, we define $\mathsf{strat}_A^{\mathcal{T}}(\ell_0 \cdot n) = l_{\mathsf{str}}(n)(A) \in \mathcal{M} \cup \{\bot\}$. For $D \subseteq \mathsf{Agt}$, we write $\mathsf{strat}_D^{\mathcal{T}}$ for $(\mathsf{strat}_A^{\mathcal{T}})_{A \in D}$.

As a first step, for each $D \subseteq \mathsf{Agt}$, we build a $\langle \mathsf{Loc}, \Sigma_{\mathcal{C}}^+ \rangle$-APT $\mathcal{A}_{\mathsf{strat}}(D)$ which will ensure that for all $A \in D$, $\mathsf{strat}_A^{\mathcal{T}}$ is *really* a strategy for player A, *i.e.*, never returns $\bot$. This automaton has only one state q_0, with $\tau(q_0, \sigma) = \bigwedge_{\ell \in \mathsf{Loc}} (\ell, q_0)$ provided that $\sigma_{\mathsf{str}}(A) \neq \bot$ for all $A \in D$. Otherwise, $\tau(q_0, \sigma) = \bot$. Finally, $\mathcal{A}_{\mathsf{strat}}$ accepts all trees having a valid execution tree (*i.e.*, Ω constantly equals 0). The following result is straightforward:

▶ **Lemma 12.** *Let $\mathcal{C}$ be a CGS, ℓ_0 be a location of $\mathcal{C}$, and $D \subseteq \mathsf{Agt}$. Let $\mathcal{T} = \langle T, l \rangle$ be a $\Sigma_{\mathcal{C}}^+$-labelled complete Loc-tree accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$. Then $\mathcal{T}$ is accepted by $\mathcal{A}_{\mathsf{strat}}(D)$ iff for each player $A \in D$, $\mathsf{strat}_A^{\mathcal{T}}$ never equals $\bot$.*

We now build an automaton for checking that proposition p_o labels outcomes of $\mathcal{T}$. More precisely, let $D \subseteq \mathsf{Agt}$ be a set of players. The automaton $\mathcal{A}_{\mathsf{out}}(D)$ will accept $\mathcal{T}$ iff

---

[2] Notice that $|\Sigma_{\mathcal{C}}| = |\mathsf{Loc}|$ and $|\Sigma_{\mathcal{C}}^+|$ is linear in the size of the input, as we assume an explicit representation of the Edg function [13].

p_o labels exactly the outcomes of strategies $\mathsf{strat}_A^{\mathcal{T}}$ for players $A \in D$. This is achieved by the following two-state automaton $\mathcal{A}_{\mathsf{out}}(D) = \langle Q, q_\in, \tau, \Omega \rangle$: $Q = \{q_\in, q_\notin\}$, $q_\in$ is the initial state, Ω constantly equals 0, and the transition function is defined as follows: if $p_o \notin \sigma$, then $\tau(q_\in, \sigma) = \bot$ and $\tau(q_\notin, \sigma) = \bigwedge_{\ell \in \mathsf{Loc}}(\ell, q_\notin)$; otheriwse, $\tau(q_\notin, \sigma) = \bot$ and

$$\tau(q_\in, \sigma) = \bigwedge_{\ell \in \mathsf{Next}(\sigma, D)} (\ell, q_\in) \wedge \bigwedge_{\ell \notin \mathsf{Next}(\sigma, D)} (\ell, q_\notin)$$

where $\mathsf{Next}(\sigma, D)$ is

$$\{\ell \in \mathsf{Loc} \mid \exists (m_i)_i \in \mathcal{M}^{\mathsf{Agt}} \text{ s.t. } (\sigma_{\mathsf{Loc}}, \ell) = \sigma_{\mathsf{Edg}}(\sigma_{\mathsf{Loc}}, (m_i)_i) \text{ and } \forall A_i \in D.\ \sigma_{\mathsf{str}}(A_i) = m_i\}.$$

In other terms, $\mathsf{Next}(\sigma, D)$ returns the set of successor states of state σ_{Loc} if players in D follow the strategies given by σ_{str}, and according to the transition table σ_{Edg}. Notice that $\mathsf{Next}(\sigma, D)$ is non-empty iff $\sigma_{\mathsf{str}}(A_i) \neq \bot$ for all $A_i \in D$. We then have:

▶ **Lemma 13.** *Let $\mathcal{C}$ be a CGS, and ℓ_0 be one of its locations, and $D \subseteq \mathsf{Agt}$. Let $\mathcal{T} = \langle T, l \rangle$ be a $\Sigma_\mathcal{C}^+$-labelled complete Loc-tree accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$ and $\mathcal{A}_{\mathsf{strat}}(D)$. Then $\mathcal{T}$ is accepted by $\mathcal{A}_{\mathsf{out}}(D)$ iff for all $n \in T$, $p_o \in l_p(n)$ iff the finite run $\ell_0 \cdot n$ is an outcome of $\mathsf{strat}_D^{\mathcal{T}}$ from ℓ_0.*

4.4 Boolean operations, projection, non-determinization, …

In this section, we review some classical results about alternating tree automata, which we will use in our construction.

▶ **Lemma 14.** *[15, 16] Let $\mathcal{A}$ and $\mathcal{B}$ be two $\langle S, \Sigma \rangle$-APTs that respectively accept the languages A and B. We can build two $\langle S, \Sigma \rangle$-APTs $\mathcal{C}$ and $\mathcal{D}$ that respectively accept the languages $A \cap B$ and $\overline{A}$ (the complement of A in the set of Σ-labelled S-trees). The size and index of $\mathcal{C}$ are at most $(|\mathcal{A}| + |\mathcal{B}|)$ and $\max(\mathsf{idx}(\mathcal{A}), \mathsf{idx}(\mathcal{B})) + 1$, while those of $\mathcal{D}$ are $|A|$ and $\mathsf{idx}(\mathcal{A})$.*

▶ **Lemma 15.** *[16] Let $\mathcal{A}$ be a $\langle S, \Sigma \rangle$-APT. We can build a $\langle S, \Sigma \rangle$-NPT $\mathcal{N}$ accepting the same language as $\mathcal{A}$, and such that $|\mathcal{N}| \in 2^{O(|\mathcal{A}|\mathsf{idx}(\mathcal{A}) \cdot \log(|\mathcal{A}|\mathsf{idx}(\mathcal{A})))}$ and $\mathsf{idx}(\mathcal{N}) \in O(|\mathcal{A}|\mathsf{idx}(\mathcal{A}))$.*

▶ **Lemma 16.** *[14] Let $\mathcal{A}$ be a $\langle S, \Sigma \rangle$-NPT, with $\Sigma = \Sigma_1 \times \Sigma_2$. For all $i \in \{1, 2\}$, we can build a $\langle S, \Sigma \rangle$-NPT $\mathcal{B}_i$ such that, for any tree $\mathcal{T}$, it holds: $\mathcal{T} \in \mathcal{L}(\mathcal{B}_i)$ iff $\exists \mathcal{T}' \in \mathcal{L}(\mathcal{A})$. $\mathsf{proj}_{\Sigma_i}(\mathcal{T}) = \mathsf{proj}_{\Sigma_i}(\mathcal{T}')$. The size and index of $\mathcal{B}_i$ are those of $\mathcal{A}$.*

▶ **Lemma 17.** *Let $\mathcal{A}$ be a $\langle S, \Sigma \times 2^{\{p\}} \rangle$-APT s.t. for any two $\Sigma \times 2^{\{p\}}$-labelled S-trees $\mathcal{T}$ and $\mathcal{T}'$ with $\mathsf{proj}_\Sigma(\mathcal{T}) = \mathsf{proj}_\Sigma(\mathcal{T}')$, we have $\mathcal{T} \in \mathcal{L}(\mathcal{A})$ iff $\mathcal{T}' \in \mathcal{L}(\mathcal{A})$. Then we can build a $\langle S, \Sigma \times 2^{\{p\}} \rangle$-APT $\mathcal{B}$ s.t. for all $\Sigma \times 2^{\{p\}}$-labelled S-tree $\mathcal{T} = \langle T, l \rangle$, it holds: $\mathcal{T} \in \mathcal{L}(\mathcal{B})$ iff $\forall n \in T.\ (p \in l(n)$ iff $\mathcal{T}_n \in \mathcal{L}(\mathcal{A}))$. Then $\mathcal{B}$ has size $O(|\mathcal{A}|)$ and index $\mathsf{idx}(\mathcal{A}) + 1$.*

4.5 Transforming an ATL_{sc} formula into an alternating tree automaton

▶ **Lemma 18.** *Let $\mathcal{C}$ be a CGS with finite state space Loc. Let ψ be an ATL_{sc}-formula, and $D \subseteq \mathsf{Agt}$ be a coalition. We can build a $\langle \mathsf{Loc}, \Sigma_\mathcal{C}^+ \rangle$-APT $\mathcal{A}_{\psi, D}$ s.t.*

▬ *for any $\Sigma_\mathcal{C}^+$-labelled complete Loc-tree $\mathcal{T}$ accepted by $\mathcal{A}_\mathcal{C}$ and by $\mathcal{A}_{\mathsf{strat}}(D)$, it holds*

$$\mathcal{T} \in \mathcal{L}(\mathcal{A}_{\psi, D}) \quad \Leftrightarrow \quad \mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_D^{\mathcal{T}}} \psi;$$

▬ *for any two $\Sigma_\mathcal{C}^+$-labelled complete Loc-tree $\mathcal{T}$ and $\mathcal{T}'$ s.t. $\mathsf{proj}_{\Sigma_\mathcal{C}'}(\mathcal{T}) = \mathsf{proj}_{\Sigma_\mathcal{C}'}(\mathcal{T}')$, with $\Sigma_\mathcal{C}' = \Sigma_\mathcal{C} \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt}}$, we have*

$$\mathcal{T} \in \mathcal{L}(\mathcal{A}_{\psi, D}) \quad \Leftrightarrow \quad \mathcal{T}' \in \mathcal{L}(\mathcal{A}_{\psi, D}).$$

The size of $\mathcal{A}_{\psi,D}$ is at most d-exponential, where d is the number of (nested) strategy quantifiers in ψ. Its index is $d-1$-exponential.

Sketch of proof. The proof proceeds by induction on the structure of formula ψ. The case of atomic propositions is straightforward. Applying Lemma 14, we immediately get the result for the case when φ is a boolean combination of subformulas.

We now sketch the proof for the case when $\psi = \langle\cdot A\cdot\rangle \mathbf{X}\,\varphi$. The case formulas $\langle\cdot A\cdot\rangle\,\varphi_1 \mathbf{U}\,\varphi_2$ and[3] $\langle\cdot A\cdot\rangle\,\varphi_1 \mathbf{R}\,\varphi_2$ could be handled similarly. The idea of the construction is as follows: we use automaton $\mathcal{A}_{\mathsf{out}}(D\cup A)$ to label outcomes with p_o, $\mathcal{A}_{\varphi,D\cup A}$ to label nodes where φ holds, and build an intermediate automaton $\mathcal{A}_f$ to check that all the outcomes satisfy $\mathbf{X}\,\varphi$. We then project out the strategy of coalition A, which yields the automaton for $\langle\cdot A\cdot\rangle \mathbf{X}\,\varphi$.

Assume that we have already built the automaton $\mathcal{A}_{\varphi,D\cup A}$ (inductively). Applying Lemma 17 to $\mathcal{A}_{\varphi,D\cup A}$ with the extra proposition p_r, we get an automaton $\mathcal{B}_{p_r,\varphi,D\cup A}$ such that, given a tree $\mathcal{T} = \langle T, l\rangle$ accepted by $\mathcal{A}_{\mathcal{C}}$ and $\mathcal{A}_{\mathsf{strat}}(D\cup A)$, it holds

$$\mathcal{T} \in \mathcal{L}(\mathcal{B}_{p_r,\varphi,D\cup A}) \quad \Leftrightarrow \quad \forall n \in T.\ (p_r \in l(n) \Leftrightarrow \mathcal{C}, l_{\mathsf{Loc}}(n) \models_{\mathsf{strat}_{D\cup A}^{\mathcal{T}_n}} \varphi). \tag{1}$$

In order to check that all the outcome satisfy $\mathbf{X}\,\varphi$, we simply have to build an automaton $\mathcal{A}_f$ for checking the $\mathsf{CTL}^\star$ property $\mathbf{A}(\mathbf{G}\,p_o \to \mathbf{X}\,p_r)$. We refer to [12] for this classical construction. This automaton $\mathcal{A}_f$ has the following property: for any $\Sigma_{\mathcal{C}}^+$-labelled Loc-tree $\mathcal{T} = \langle T, l\rangle$, we have

$$\mathcal{T} \in \mathcal{L}(\mathcal{A}_f) \quad \Leftrightarrow \quad \mathcal{T}, \varepsilon \models \mathbf{A}(\mathbf{G}\,p_o \to \mathbf{X}\,p_r). \tag{2}$$

Now, let $\mathcal{H}$ be the product of $\mathcal{A}_{\mathsf{strat}}(A)$, $\mathcal{A}_{\mathsf{out}}(D\cup A)$, $\mathcal{A}_f$ and $\mathcal{B}_{p_r,\varphi,D\cup A}$, and let $\mathcal{T}$ be a tree accepted by $\mathcal{A}_{\mathcal{C}}$ and $\mathcal{A}_{\mathsf{strat}}(D)$. If $\mathcal{T}$ is accepted by $\mathcal{H}$, then $D\cup A \subseteq \mathsf{dom}(\mathcal{T})$ and all the outcomes of the strategy $\mathsf{strat}_{D\cup A}^{\mathcal{T}}$ from $l_{\mathsf{Loc}}(\varepsilon)$ satisfy $\mathbf{X}\,\varphi$.

The converse does not hold in general, but we prove a weaker form: from $\mathcal{T} = \langle T, l\rangle$, accepted by $\mathcal{A}_{\mathcal{C}}$ and $\mathcal{A}_{\mathsf{strat}}(D)$, and such that $D\cup A \subseteq \mathsf{dom}(\mathcal{T})$ and the outcomes of $\mathsf{strat}_{D\cup A}^{\mathcal{T}}$ from $l_{\mathsf{Loc}}(\varepsilon)$ satisfy $\mathbf{X}\,\varphi$, we build $\mathcal{T}' = \langle T, l'\rangle$ such that $\mathsf{proj}_{\Sigma_{\mathcal{C}}'}(\mathcal{T}) = \mathsf{proj}_{\Sigma_{\mathcal{C}}'}(\mathcal{T}')$, and $\mathcal{T}'$ is accepted by $\mathcal{H}$. To do this, it suffices to modify the labelling of $\mathcal{T}$ with p_o and p_r, in such a way that $\mathcal{T}'$ is accepted by $\mathcal{A}_{\mathsf{out}}(D\cup A)$ and $\mathcal{B}_{p_r,\varphi,D\cup A}$. This ensures that $\mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_{D\cup A}^{\mathcal{T}'}} \langle\cdot\varnothing\cdot\rangle \mathbf{X}\,\varphi$, and that $\mathcal{T}'$ is also accepted by $\mathcal{A}_f$. In the end, we have that for any tree $\mathcal{T} = \langle T, l\rangle$ accepted by $\mathcal{A}_{\mathcal{C}}$ and $\mathcal{A}_{\mathsf{strat}}(D)$,

$$D\cup A \subseteq \mathsf{dom}(\mathcal{T}) \quad \text{and} \quad \mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_{D\cup A}^{\mathcal{T}}} \langle\cdot\varnothing\cdot\rangle \mathbf{X}\,\varphi \qquad \Leftrightarrow$$
$$\exists \mathcal{T}' \text{ s.t. } \mathsf{proj}_{\Sigma_{\mathcal{C}}'}(\mathcal{T}') = \mathsf{proj}_{\Sigma_{\mathcal{C}}'}(\mathcal{T}) \text{ and } T' \in \mathcal{L}(\mathcal{H}). \tag{3}$$

Applying Lemma 15, we get a $\langle\Sigma_{\mathcal{C}}^+, \mathsf{Loc}\rangle$-NPT $\mathcal{N}$ such that $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{H})$. We can then apply Lemma 16 for $\Sigma_{\mathcal{C}}^+ = (\Sigma_{\mathcal{C}} \times (\mathcal{M}\cup\{\bot\})^{\mathsf{Agt}\smallsetminus A}) \times ((\mathcal{M}\cup\{\bot\})^A \times 2^{p_o,p_l,p_r})$ on the NPT $\mathcal{N}$; the resulting $\langle\Sigma_{\mathcal{C}}^+, \mathsf{Loc}\rangle$-NPT $\mathcal{P}$ accepts all trees $\mathcal{T}$ whose labelling on $(\mathcal{M}\cup\{\bot\})^A \times 2^{p_o,p_l,p_r}$ can be modified in order to have the tree accepted by $\mathcal{N}$. Then $\mathcal{P}$ satisfies both properties of the Lemma: the second property directly follows from the use Lemma 16. For the first one, pick $\mathcal{T} = \langle T, l\rangle$ accepted by $\mathcal{A}_{\mathcal{C}}$ and by $\mathcal{A}_{\mathsf{strat}}(D)$. If $\mathcal{T}$ is accepted by $\mathcal{P}$, then from Lemma 16, there exists a tree $\mathcal{T}' = \langle T, l'\rangle$, with the same labelling as $\mathcal{T}$ on $\Sigma_{\mathcal{C}} \times (\mathcal{M}\cup\{\bot\})^{\mathsf{Agt}\smallsetminus A}$, and accepted by $\mathcal{N}$. Since $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{H})$, and from (3), we get that $D\cup A \subseteq \mathsf{dom}(\mathcal{T}')$ and

---

[3] The "release" modality $\mathbf{R}$ is the dual of $\mathbf{U}$, defined by $\varphi_1 \mathbf{R}\,\varphi_2 \equiv \neg[(\neg\varphi_1) \mathbf{U}\,(\neg\varphi_2)]$. Notice that $\mathbf{X}$ is self-dual as we only evaluate formulas along infinite outcomes.

$\mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_{D\cup A}^{\mathcal{T}'}} \langle\varnothing\rangle\, \mathbf{X}\,\varphi$. Thus $\mathsf{strat}_A^{\mathcal{T}'}$ is a strategy for coalition A, and it witnesses the fact that $\mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_D^{\mathcal{T}'}} \langle A\rangle\, \mathbf{X}\,\varphi$, and we get the desired result since $\mathsf{strat}_D^{\mathcal{T}} = \mathsf{strat}_D^{\mathcal{T}'}$. Conversely, if $\mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_D^{\mathcal{T}}} \langle A\rangle\, \mathbf{X}\,\varphi$, then we can modify the labelling of $\mathcal{T}$ with a witnessing strategy for A, obtaining a tree $\mathcal{T}'$ such that $\mathcal{C}, l_{\mathsf{Loc}}(\varepsilon) \models_{\mathsf{strat}_{D\cup A}^{\mathcal{T}'}} \langle\varnothing\rangle\, \mathbf{X}\,\varphi$. From (3), $\mathcal{T}'$ can in turn be modified into a tree $\mathcal{T}''$, with $\mathsf{proj}_{\Sigma_{\mathcal{C}}'}(\mathcal{T}'') = \mathsf{proj}_{\Sigma_{\mathcal{C}}'}(\mathcal{T}')$, in such a way that $\mathcal{T}'' \in \mathcal{L}(\mathcal{H})$. Finally, since the projections of $\mathcal{T}''$ and $\mathcal{T}$ coincide on $(\Sigma_{\mathcal{C}} \times (\mathcal{M} \cup \{\bot\})^{\mathsf{Agt}\smallsetminus A})$, it holds that $\mathcal{T}$ is accepted by $\mathcal{P}$. This concludes the proof for $\langle A\rangle\, \mathbf{X}\,\varphi$.

The proofs for the "until" and "release" modalities follow the same lines, using p_l and p_r as extra atmomic propositions for the left- and right-hand subformulas, and modifying automaton $\mathcal{A}_f$ so that it accepts trees satisfying $\mathbf{A}(\mathbf{G}\,p_o \rightarrow p_l\,\mathbf{U}\,p_r)$ and $\mathbf{A}(\mathbf{G}\,p_o \rightarrow p_l\,\mathbf{R}\,p_r)$, respectively. Finally, when $\psi = \rangle A\langle\,\varphi$, we let $\mathcal{A}_{\rangle A\langle\,\varphi, D} = \mathcal{A}_{\varphi, D\smallsetminus A}$, which is easily proved to satisfy both requirements.

Unless $A = \varnothing$, the construction of the automaton for $\langle A\rangle\, \mathbf{X}\,\varphi$ (or $\langle A\rangle\,\varphi_1\,\mathbf{U}\,\varphi_2$ or $\langle A\rangle\,\varphi_1\,\mathbf{R}\,\varphi_2$) involves an exponential blowup in the size and index of the automata for the subformulas, and the index is bilinear in the size and index of these automata. In the end, for a formula involving d nested non-empty strategy quantifiers, the automaton has size d-exponential and index $d - 1$-exponential. ◄

▶ **Corollary 19.** *Given an ATL_{sc} formula φ, a CGS $\mathcal{C}$ and a state ℓ_0 of $\mathcal{C}$, we can built an alternating parity tree automaton $\mathcal{A}$ s.t. $\mathcal{L}(\mathcal{A}) \neq \varnothing$ iff $\mathcal{C}, \ell_0 \models_\varnothing \varphi$. Moreover, $\mathcal{A}$ has size d-exponential and index $d - 1$-exponential, where d is the number of nested non-empty strategy quantifiers.*

Proof. It suffices to take the product of the automaton $\mathcal{A}_{\varphi, \varnothing}$ (from Lemma 18) with $\mathcal{A}_{\mathcal{C}, \ell_0}$. In case this $\langle\mathsf{Loc}, \Sigma_{\mathcal{C}}^+\rangle$-APT accepts a tree $\mathcal{T}$, Lemma 18 entails that $\mathcal{C}, \ell_0 \models_\varnothing \varphi$. Conversely, if $\mathcal{C}, \ell_0 \models \varphi$, then the extended unwinding tree $\mathcal{T} = \langle T, l\rangle$ of $\mathcal{C}$ from ℓ_0 in which $l_{\mathsf{str}}(n) = \bot$ for all $n \in T$ is accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$ (and, trivially, by $\mathcal{A}_{\mathsf{strat}}(\varnothing)$), and from Lemma 18, it is also accepted by $\mathcal{A}_{\varphi, \varnothing}$. ◄

Proof of Theorem 9. The first statement directly follows from the previous corollary, since emptiness of alternating parity tree automata $\mathcal{A}$ can be checked in time $\exp(O(|\mathcal{A}| \times \mathsf{idx}(\mathcal{A})))$.

For the second statement, notice that the size and index of $\mathcal{A}_{\varphi, \varnothing}$ in the proof of Corollary 19 do not depend on the CGS $\mathcal{C}$. Hence the automaton $\mathcal{A}$ of Corollary 19 has size linear in $|\mathcal{C}|$, and can be computed in time exponential in $|\mathcal{C}|$ (because $\mathcal{A}_{out}(D)$ requires the computation of $\mathsf{Next}(\sigma, D)$). Non-emptiness is then checked in time exponential in $|\mathcal{C}|$. ◄

▶ Remark. Our algorithm can easily be modified in order to handle $\mathsf{ATL}_{sc}^\star$. One solution is to rely on Theorem 7, but our translation from $\mathsf{ATL}_{sc}^\star$ to ATL_{sc} may double the number of nested non-empty strategy quantifiers. The algorithm would then be in $(2k + 1)$-EXPTIME, where k is the number of nested strategy quantifications. Another solution is to adapt our construction, by replacing each state-subformula with a fresh atomic proposition, and build the automaton $\mathcal{A}_f$ for a more complex $\mathsf{CTL}^\star$ formula. This results in a $(k + 1)$-EXPTIME algorithm. In both cases, the program complexity is unchanged, in EXPTIME.

Similarly, our algorithm could be modified to handle strategy logic [9]. One important difference is that strategy logic may require to store several strategies per player in the tree, while ATL_{sc} only stores one strategy per player. This would then be reflected in a modified version of the Next function we use when building $\mathcal{A}_{\mathsf{out}}(D)$, where we should also indicate which strategies we use for which player.

5   Conclusions

Strategy contexts provide a very expressive extension of the semantics of ATL, as we witnessed by the fact that ATL_{sc} and $\mathsf{ATL}_{sc}^\star$ are equally expressive. We also designed a tree-automata-based algorithm for model-checking both logics on the whole class of CGSs, based on a novel encoding of strategies as a tree.

Our algorithms involve a non-elementary blowup in the size of the formula, which we currently don't know if it can be avoided. Trying to establish lower-bounds on the complexity of the problems is part of our future works. Regarding expressiveness, ATL_{sc} can distinguish between alternating-bisimilar CGSs, and we are also looking for a behavioural equivalence that could characterize the distinguishing power of ATL_{sc}.

────── **References** ──────

1   T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-time temporal logics with irrevocable strategies. In *TARK'07*, p. 15–24, 2007.

2   T. Ågotnes, V. Goranko, and W. Jamroga. Strategic commitment and release in logics for multi-agent systems. Tech. Rep. 08-01, Clausthal U. of Technology, Germany, 2008.

3   R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *COMPOS'97*, LNCS 1536, p. 23–60. Springer, 1998.

4   R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

5   R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In *CONCUR'98*, LNCS 1466, p. 163–178. Springer, 1998.

6   Ch. Baier, T. Brázdil, M. Größer, and A. Kučera. Stochastic game logic. In *QEST'07*, p. 227–236. IEEE Comp. Soc., 2007.

7   Th. Brihaye, A. Da Costa, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In *LFCS'09*, LNCS 5407, p. 92–106. Springer, 2009.

8   K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In *CONCUR'07*, LNCS 4703, p. 59–73. Springer, 2007.

9   K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Inf. & Comp.*, 208(6):677–693, 2010.

10   E. M. Clarke and E. A. Emerson. Design and synthesis of synchrization skeletons using branching-time temporal logic. In *LOP'81*, LNCS 131, p. 52–71. Springer, 1982.

11   A. Da Costa, F. Laroussinie, and N. Markey. Expressiveness and decidability of ATL with strategy contexts. Tech. Rep. LSV-10-14, Lab Spécification & Vérification, France, 2010.

12   O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. *J. ACM*, 47(2):312–360, 2000.

13   F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *LMCS*, 4(2), 2008.

14   D. E. Muller and P. E. Schupp. Alternating automata on infinite objects, determinacy and Rabin's theorem. In *EPIT'84*, LNCS 192, p. 99–107. Springer, 1985.

15   D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *TCS*, 54(2-3):267–276, 1987.

16   D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *TCS*, 141(1-2):69–107, 1995.

17   S. Pinchinat. A generic constructive solution for concurrent games with expressive constraints on strategies. In *ATVA'07*, LNCS 4762, p. 253–267. Springer, 2007.

18   A. Pnueli. The temporal logic of programs. In *FOCS'77*, p. 46–57. IEEE Comp. Soc., 1977.

19 J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *SOP'82*, LNCS 137, p. 337–351. Springer, 1982.

20 D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In *TARK'07*, p. 269–278, 2007.