

Expressiveness and decidability of ATL with strategy contexts

Arnaud Da Costa, François Laroussinie,
Nicolas Markey

July 2010

Research report LSV-10-14



LSV

Laboratoire Spécification & Vérification

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Expressiveness and decidability of ATL with strategy contexts

Arnaud Da Costa¹, François Laroussinie², and Nicolas Markey¹

¹Lab. Spécification & Vérification, ENS Cachan & CNRS, France
{dacosta,markey}@lsv.ens-cachan.fr

²LIAFA, Univ. Paris Diderot - Paris 7 & CNRS, France
Francoisl@liafa.jussieu.fr

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_{sc}^* , equally expressive: any formula in ATL_{sc}^* can be translated into an equivalent, linear-size ATL_{sc} formula. Despite the high expressiveness of these logics, we prove that they remain decidable by designing a tree-automata-based algorithm for model-checking ATL_{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 [17, 10, 18]. 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 states) 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

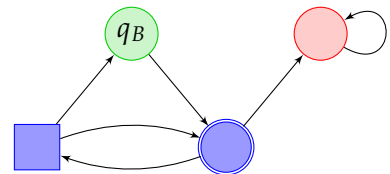


Figure 1: Example of a two-player turn-based game

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 obviously follows that Player A has no strategy to keep the system in safe states.

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), 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 ATL_{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 ATL_{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 ATL^* .

- *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 $\exists \sigma_A. [\mathbf{G} (\exists \sigma_B. (\mathbf{GF} q_B) (\sigma_A, \sigma_B))] (\sigma_A)$. 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.
- $QD\mu$ [16] 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_2^P -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 [19], an NP algorithm is proposed for the same subclass, but where strategy quantification is restricted to memoryless strategies.

2 ATL with strategy contexts

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 $\langle \text{Loc}, \text{Lab}, \delta, \text{Agt}, \mathcal{M}, \text{Mov}, \text{Edg} \rangle$ where:

- $\langle \text{Loc}, \text{Lab}, \delta \rangle$ is a (possibly infinite) Kripke structure, where Loc is the set of locations, $\text{Lab}: \text{Loc} \rightarrow 2^{AP}$ is a labelling function, and $\delta \subseteq \text{Loc} \times \text{Loc}$ is the set of transitions;
- $\text{Agt} = \{A_1, \dots, A_p\}$ is a finite set of agents (or players);
- \mathcal{M} is a finite, non-empty set of moves;
- $\text{Mov}: \text{Loc} \times \text{Agt} \rightarrow \mathcal{P}(\mathcal{M}) \setminus \{\emptyset\}$ defines the (finite) set of possible moves of each agent in each location.
- $\text{Edg}: \text{Loc} \times \mathcal{M}^{\text{Agt}} \rightarrow \delta$ 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 defined as $|\text{Loc}| + |\text{Edg}|$, where $|\text{Edg}|$ is the size of the transition table*.

The intended behaviour is as follows [4]: in a location ℓ , each player A_i in Agt chooses one among her possible moves m_i in $\text{Mov}(\ell, A_i)$; the next transition to be fired is given by $\text{Edg}(\ell, (m_1, \dots, m_p))$. We write $\text{Next}(\ell)$ for the set of all transitions corresponding to possible moves from ℓ , and $\text{Next}(\ell, A_j, m_j)$, with $m_j \in \text{Mov}(\ell, A_j)$, for the restriction of $\text{Next}(\ell)$ 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 $A \subseteq \text{Agt}$ and $\ell \in \text{Loc}$, $\text{Mov}(\ell, A)$ 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 $m = (m_a)_{a \in A} \in \text{Mov}(\ell, A)$, we let $\text{Next}(\ell, A, m)$ denote the restriction of $\text{Next}(\ell)$ 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 $\rho = \ell_0 \ell_1 \dots$ of locations such that for any i , $\ell_{i+1} \in \text{Next}(\ell_i)$. Finite paths are also called *history*. The length of a history $\rho = \ell_0 \ell_1 \dots \ell_n$ is n . We write $\rho^{i \rightarrow j}$ for the part of ρ between ℓ_i and ℓ_j (inclusive). In particular, $\rho^{i \rightarrow j}$ is empty iff $j < i$. We simply write ρ^i for $\rho^{i \rightarrow i}$, denoting the $i + 1$ -st location ℓ_i of ρ . We also define $\text{first}(\rho) = \rho^0$, and, if ρ has finite length n , $\text{last}(\rho) = \rho^n$. Given a history π of length n and a path ρ such that $\text{last}(\pi) = \text{first}(\rho)$, the concatenation of π and ρ is the path $\tau = \pi \cdot \rho$ such that $\tau^{0 \rightarrow n} = \pi$ and $\tau^{n \rightarrow \infty} = \rho$ (notice that the last location of π and the first location of ρ are “merged”).

A *strategy* for a player $A_i \in \text{Agt}$ is a function f_i that maps any history to a possible move for A_i , i.e., satisfying $f_i(\ell_0 \dots \ell_m) \in \text{Mov}(\ell_m, A_i)$. A strategy for a coalition A of agents

*Our results would still hold (with the same complexity) if we consider symbolic CGSs [12], where the transition table is encoded through boolean formulas.

is a mapping assigning a strategy to each agent in the coalition. The set of strategies for A is denoted $\text{Strat}(A)$. The *domain* of $F_A \in \text{Strat}(A)$ (denoted $\text{dom}(F_A)$) is A . Given a coalition B , the strategy $(F_A)|_B$ (resp. $(F_A)_{\setminus B}$) denotes the restriction of F_A to the coalition $A \cap B$ (resp. $A \setminus B$).

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 $\text{Out}(\rho, F_A)$: a path $\pi = \rho \cdot \ell_1 \ell_2 \dots$ is in $\text{Out}(\rho, F_A)$ iff, writing $\ell_0 = \text{last}(\rho)$, for all $i \geq 0$ there exists a set of moves $(m_k^i)_{A_k \in \text{Agt}}$ such that

- $m_k^i \in \text{Mov}(\ell_i, A_k)$ for all $A_k \in \text{Agt}$,
- $m_k^i = f_{A_k}(\pi^{0 \rightarrow n+i})$ if $A_k \in A$,
- $\ell_{i+1} \in \text{Next}(\ell_i, \text{Agt}, (m_k^i)_{A_k \in \text{Agt}})$.

We write $\text{Out}^\infty(\rho, F_A)$ for the set of infinite outcomes of F_A after ρ . Note that $\text{Out}(\rho, F_A) \subseteq \text{Out}(\rho, (F_A)|_B)$ for any two coalitions A and B , and that $\text{Out}(\rho, F_\emptyset)$ represents the set of all paths starting with ρ .

It is also possible to *combine* two strategies $F \in \text{Strat}(A)$ and $F' \in \text{Strat}(B)$, resulting in a strategy $F \circ F' \in \text{Strat}(A \cup B)$ defined as follows:

$$(F \circ F')|_{A_j}(\rho) = \begin{cases} F|_{A_j}(\rho) & \text{if } A_j \in A \\ F'|_{A_j}(\rho) & \text{if } A_j \in B \setminus A. \end{cases}$$

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 $\text{last}(\rho) = \text{first}(\pi)$, as $F^\rho(\pi) = F(\rho \cdot \pi)$.

Alternating-time temporal logics. The logics ATL and ATL* have been defined in [4] as extensions of CTL and CTL* with strategy quantification. Following [7], we further extend them with *strategy contexts*:

DEFINITION 2. *The syntax of ATL_{sc}^* is defined by the following grammar:*

$$\begin{aligned} \text{ATL}_{sc}^* \ni \varphi_s, \psi_s &::= p \mid \neg \varphi_s \mid \varphi_s \vee \psi_s \mid \langle A \rangle \varphi_p \mid \cdot \rangle A \langle \varphi_s \\ \varphi_p, \psi_p &::= \varphi_s \mid \neg \varphi_p \mid \varphi_p \vee \psi_p \mid \mathbf{X} \varphi_p \mid \varphi_p \mathbf{U} \psi_p \end{aligned}$$

with $p \in \text{AP}$ and $A \subseteq \text{Agt}$. Formulas defined as φ_s are called *state formulas*, while φ_p defines *path formulas*.

That a formula φ in 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 $\text{Strat}(A)$ for a coalition A), denoted $\mathcal{C}, \rho \models_F \varphi$, is defined as follows:

$$\begin{aligned} \mathcal{C}, \rho \models_F p &\text{ iff } p \in \text{Lab}(\text{first}(\rho)), \\ \mathcal{C}, \rho \models_F \neg \varphi &\text{ iff } \mathcal{C}, \rho \not\models_F \varphi, \\ \mathcal{C}, \rho \models_F \varphi \vee \psi &\text{ iff } \mathcal{C}, \rho \models_F \varphi \text{ or } \mathcal{C}, \rho \models_F \psi, \\ \mathcal{C}, \rho \models_F \langle A \rangle \varphi_p &\text{ iff } \exists F_A \in \text{Strat}(A). \forall \rho' \in \text{Out}^\infty(\text{first}(\rho), F_A \circ F). \\ &\hspace{15em} \mathcal{C}, \rho' \models_{F_A \circ F} \varphi_p, \end{aligned}$$

$$\begin{aligned}
\mathcal{C}, \rho \models_F \cdot A \langle \varphi_s \rangle & \text{ iff } \mathcal{C}, \rho \models_{F \setminus A} \varphi_s, \\
\mathcal{C}, \rho \models_F \mathbf{X} \varphi_p & \text{ iff } \mathcal{C}, \rho^{1 \rightarrow \infty} \models_{F \rho^{0 \rightarrow 1}} \varphi_p, \\
\mathcal{C}, \rho \models_F \varphi_p \mathbf{U} \psi_p & \text{ iff } \exists i \geq 0. \mathcal{C}, \rho^{i \rightarrow \infty} \models_{F \rho^{0 \rightarrow i}} \psi_p \text{ and} \\
& \forall 0 \leq j < i. \mathcal{C}, \rho^{j \rightarrow \infty} \models_{F \rho^{0 \rightarrow j}} \varphi_p.
\end{aligned}$$

As stated in the following lemma, 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 $\mathcal{C}, \text{first}(\rho) \models_F \varphi_s$ when it raises no ambiguity):

LEMMA 3. *Let \mathcal{C} be a CGS, and $F \in \text{Strat}(A)$ be a strategy context. For any state formula φ_s , and for any two infinite paths ρ and ρ' with $\text{first}(\rho) = \text{first}(\rho')$, it holds*

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

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 $\cdot A \langle \cdot \rangle$ 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. \square

The logic ATL_{sc} is obtained by restricting ATL_{sc}^* to the following grammar:

$$\begin{aligned}
\text{ATL}_{sc} \ni \varphi_s, \psi_s & ::= p \mid \neg \varphi_s \mid \varphi_s \vee \psi_s \mid \langle A \rangle \varphi_p \mid \cdot A \langle \varphi_s \rangle \\
\varphi_p, \psi_p & ::= \neg \varphi_p \mid \mathbf{X} \varphi_s \mid \varphi_s \mathbf{U} \psi_s
\end{aligned}$$

We define the following shorthands, which will be useful in the sequel:

$$\begin{aligned}
\top & \stackrel{\text{def}}{=} p \vee \neg p & \mathbf{F} \varphi & \stackrel{\text{def}}{=} \top \mathbf{U} \varphi & \langle A \rangle \varphi_s & \stackrel{\text{def}}{=} \langle A \rangle (\perp \mathbf{U} \varphi_s) \\
\perp & \stackrel{\text{def}}{=} \neg \top & \mathbf{G} \varphi & \stackrel{\text{def}}{=} \neg \mathbf{F} \neg \varphi & \langle\langle A \rangle\rangle \varphi & \stackrel{\text{def}}{=} \cdot \text{Agt} \langle \langle A \rangle \varphi \rangle
\end{aligned}$$

The last shorthand corresponds to ATL^* strategy quantifier (provided that the subformula φ is in ATL^*). Thanks to the previous shorthand, strategy quantifiers can be followed by both path and state quantifiers even in ATL_{sc} . It follows that ATL and ATL^* can be translated into ATL_{sc} and ATL_{sc}^* resp. However, it must be noted 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 $\langle A \rangle \mathbf{G} (\langle \emptyset \rangle \mathbf{F} P \wedge \langle \emptyset \rangle \mathbf{F} P')$ is equivalent in a standard Kripke structure (seen as a CGS with one single player A) to the CTL^* formula $\text{E}(\overline{\mathbf{F}} P \wedge \overline{\mathbf{F}} P')$ 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. ATL_{sc}^*) is strictly more expressive than ATL (resp. ATL^*). Game Logic (see [4]) can also be translated into ATL_{sc}^* (while the converse is not true). In this section, we present some new results on the expressiveness of ATL_{sc} .

Alternating bisimulation. Strategy contexts also increase the *distinguishing power* of alternating logics. Indeed, logics like ATL, ATL^{*}, GL or AMC cannot distinguish between two games that are alternating-bisimilar [5]. Formally this behavioral equivalence is defined as follows: Given two CGSs \mathcal{C} and \mathcal{C}' , a relation $R \subseteq \text{Loc}_{\mathcal{C}} \times \text{Loc}_{\mathcal{C}'}$ is an *alternating bisimulation* when for any $(\ell, \ell') \in R$, we have:

- $\text{Lab}_{\mathcal{A}}(\ell) = \text{Lab}_{\mathcal{B}}(\ell')$,
- for any coalition $A \subseteq \text{Agt}$, it holds

$$\begin{aligned} \forall m_1 \in \text{Mov}_{\mathcal{A}}(\ell, A). \exists m'_1 \in \text{Mov}_{\mathcal{B}}(\ell', A). \\ \forall m'_2 \in \text{Mov}_{\mathcal{B}}(\ell', \text{Agt} \setminus A). \exists m_2 \in \text{Mov}_{\mathcal{A}}(\ell, \text{Agt} \setminus A) \text{ s.t.} \\ (\text{Next}(\ell, m_1 \circ m_2), \text{Next}(\ell', m'_1 \circ m'_2)) \in R, \end{aligned}$$

and symmetrically (from ℓ').

Two locations ℓ and ℓ' are said to be alternating-bisimilar when there exists an alternating-bisimulation R containing (ℓ, ℓ') .

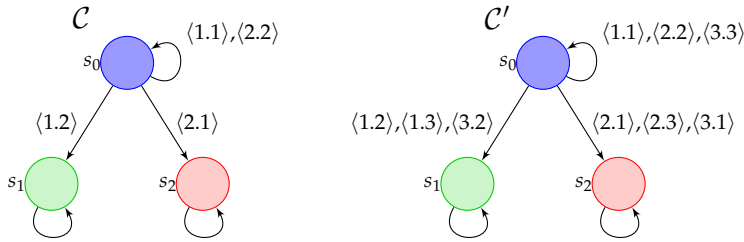


Figure 2: \mathcal{C} and \mathcal{C}' are alternating-bisimilar but can be distinguished by ATL_{sc}

It turns out that ATL_{sc} and ATL_{sc}^{*} can distinguish alternating-bisimilar locations: consider the CGSs in Figure 2; it can be seen that s_0 and s'_0 are alternating-bisimilar (the only difference between both structures is the third move in \mathcal{C}' , but it can be simulated by moves 1 or 2 in \mathcal{C} since every successor for these

two choices is also a successor of move 3 in \mathcal{C}'). But on the other hand, s'_0 satisfies $\Phi \stackrel{\text{def}}{=} \langle A_1 \cdot \rangle (\langle A_2 \cdot \rangle \mathbf{X} \text{green} \wedge \langle A_2 \cdot \rangle \mathbf{X} \text{red})$ (by letting Player A_1 play move 3), while it is rapidly checked that s_0 fails to satisfy this property. In particular, ATL and ATL^{*} cannot distinguish between these two structures.

Relative expressiveness of ATL_{sc} and ATL_{sc}^{*}. We now prove that strategy contexts bring ATL_{sc} to the same expressiveness as ATL_{sc}^{*}. This can already be observed in the following simple example: consider the CTL^{*} formula $\varphi \stackrel{\text{def}}{=} \mathbf{EG F} a$ (expressing that a occurs infinitely often along a run); it is well-known that φ cannot be expressed in CTL (where every temporal modality has to be in the immediate scope of a path quantifier). But φ is clearly equivalent (over Kripke structures, *i.e.* one-player CGSs) to the ATL_{sc} formula $\langle A \cdot \rangle \mathbf{G} \langle \emptyset \cdot \rangle \mathbf{F} a$: indeed, as soon as the strategy context is fixed for Player A , the use of $\langle \emptyset \cdot \rangle$ does not change the choice of the underlying run where $\mathbf{F} a$ is interpreted. We can extend this approach to any ATL_{sc}^{*} formula: the idea is to

1. first use *full* strategy contexts (by adding universally quantified strategies) in order to be able to insert the $\langle \emptyset \cdot \rangle$ 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. For this last point, we replace $\langle \cdot A \cdot \rangle \varphi$ with the subformula $\langle \cdot A \cdot \rangle \neg \langle \text{Agt} \setminus (A \cup B) \cdot \rangle \neg \varphi$ where B is the part of the context that A can rely on to choose her strategy (i.e., A only “knows” the strategies chosen by agents in B). This mechanism is formally described below.

Now we give the translation from ATL_{sc}^* to ATL_{sc} . Given an ATL_{sc}^* formula Φ and a coalition B , we define $\widehat{\Phi}^{[B]}$ inductively as follows:

$$\begin{array}{lll} \widehat{p}^{[B]} \stackrel{\text{def}}{=} p & \widehat{\neg \varphi}^{[B]} \stackrel{\text{def}}{=} \neg \widehat{\varphi}^{[B]} & \widehat{\varphi \wedge \psi}^{[B]} \stackrel{\text{def}}{=} \widehat{\varphi}^{[B]} \wedge \widehat{\psi}^{[B]} \\ \widehat{\mathbf{X} \varphi}^{[B]} \stackrel{\text{def}}{=} \langle \cdot \emptyset \cdot \rangle \mathbf{X} \widehat{\varphi}^{[B]} & & \widehat{\varphi \mathbf{U} \psi}^{[B]} \stackrel{\text{def}}{=} \langle \cdot \emptyset \cdot \rangle (\widehat{\varphi}^{[B]} \mathbf{U} \widehat{\psi}^{[B]}) \\ \widehat{\langle \cdot A \cdot \rangle \varphi}^{[B]} \stackrel{\text{def}}{=} \langle \cdot A \cdot \rangle \neg \langle \text{Agt} \setminus (A \cup B) \cdot \rangle \neg \widehat{\varphi}^{[A \cup B]} & & \widehat{\cdot A \langle \cdot \varphi \cdot \rangle}^{[B]} \stackrel{\text{def}}{=} \widehat{\varphi}^{[B \setminus A]} \end{array}$$

Clearly, $\widehat{\Phi}^{[B]}$ is an ATL_{sc} formula. First, we have the following lemma stating that the truth value of a state formula $\widehat{\varphi}^A$ interpreted in a strategy context H depends only on $H|_A$:

LEMMA 4. *For any state formula φ , any strategy contexts F, G and G' such that $\text{dom}(F) \cap \text{dom}(G') = \emptyset$, $\text{dom}(G) \subseteq \text{dom}(G')$ and $G'|_{\text{dom}(G)} = G$, we have:*

$$\mathcal{C}, \ell \models_{G \circ F} \widehat{\varphi}^{[\text{dom}(F)]} \Leftrightarrow \mathcal{C}, \ell \models_{G' \circ F} \widehat{\varphi}^{[\text{dom}(F)]}$$

PROOF. The proof is done by structural induction over the formula. The cases of atomic propositions and Boolean operators are straightforward.

Now consider $\varphi \stackrel{\text{def}}{=} \langle \cdot A \cdot \rangle \psi$. Then $\mathcal{C}, \ell \models_{G \circ F} \widehat{\langle \cdot A \cdot \rangle \psi}^{[\text{dom}(F)]}$ entails $\mathcal{C}, \ell \models_{G \circ F} \langle \cdot A \cdot \rangle \neg \langle \text{Agt} \setminus A \cup \text{dom}(F) \cdot \rangle \neg \widehat{\psi}^{[\text{dom}(F) \cup A]}$. Thus there exists $F_A \in \text{Strat}(A)$ such that for any $F' \in \text{Strat}(\text{Agt} \setminus \text{dom}(F) \cup A)$, we have $\mathcal{C}, \pi \models_{F' \circ F_A \circ G \circ F} \widehat{\psi}^{[\text{dom}(F) \cup A]}$ where π is the unique path in $\text{Out}(\ell, F' \circ F_A \circ G \circ F)$. Then we clearly have $\mathcal{C}, \pi \models_{F' \circ F_A \circ G' \circ F} \widehat{\psi}^{[\text{dom}(F) \cup A]}$ because $F' \circ F_A$ overwrites G and G' . The converse direction follows the same steps.

Assume $\varphi \stackrel{\text{def}}{=} \cdot A \langle \cdot \psi \cdot \rangle$. Then $\mathcal{C}, \ell \models_{G \circ F} \widehat{\varphi}^{[\text{dom}(F)]}$ entails $\mathcal{C}, \ell \models_{G \circ F} \widehat{\psi}^{[\text{dom}(F) \setminus A]}$ and we have $\mathcal{C}, \ell \models_{G' \circ F} \widehat{\psi}^{[\text{dom}(F) \setminus A]}$ by induction hypothesis (ψ is a state formula). Thus we have $\mathcal{C}, \ell \models_{G' \circ F} \widehat{\varphi}^{[\text{dom}(F)]}$. \square

Now we have the following lemma relating the truth value of φ and $\widehat{\varphi}^{[B]}$:

LEMMA 5. *Let \mathcal{C} be a CGS, ℓ be one of its locations, and F be a strategy context. Then for any ATL_{sc}^* formula φ , for any strategy context \bar{F} s.t. $\text{dom}(\bar{F}) = \text{Agt} \setminus \text{dom}(F)$, and for any outcome $\pi \in \text{Out}^\infty(\ell, \bar{F} \circ F)$, it holds*

$$\mathcal{C}, \pi \models_F \varphi \Leftrightarrow \mathcal{C}, \pi \models_{\bar{F} \circ F} \widehat{\varphi}^{[\text{dom}(F)]}.$$

Moreover, if φ is a state formula, the above result extends to any strategy context \bar{F} s.t. $\text{dom}(\bar{F}) \cap \text{dom}(F) = \emptyset$.

PROOF. We prove the result by induction on the structure of φ . The cases of atomic propositions and boolean connectives are straightforward.

- If $\varphi = \mathbf{X}\psi$: for any G s.t. $\text{dom}(\bar{F}) = \text{Agt} \setminus \text{dom}(F)$, and for the corresponding outcome π from ℓ , we have the following equivalences:

$$\begin{aligned}
\mathcal{C}, \pi \models_F \varphi &\Leftrightarrow \mathcal{C}, \pi^{1 \rightarrow \infty} \models_{F\pi^{0 \rightarrow 1}} \psi \\
&\Leftrightarrow \mathcal{C}, \pi^{1 \rightarrow \infty} \models_{(G \circ F)\pi^{0 \rightarrow 1}} \widehat{\psi}^{[\text{dom}(F)]} && \text{(by i.h.)} \\
&\Leftrightarrow \mathcal{C}, \pi \models_{G \circ F} \mathbf{X}\widehat{\psi}^{[\text{dom}(F)]} \\
&\Leftrightarrow \mathcal{C}, \pi \models_{G \circ F} \langle \cdot \emptyset \cdot \rangle \mathbf{X}\widehat{\psi}^{[\text{dom}(F)]} \\
&\hspace{15em} \text{(because } \text{Out}^\infty(\ell, G \circ F) = \{\pi\}\text{)}
\end{aligned}$$

- If $\varphi = \psi_1 \mathbf{U} \psi_2$: this case can be handled in a similar way as for the previous case, and we omit it.
- If $\varphi = \langle \cdot A \cdot \rangle \psi$: we prove the second, more general statement. Let G be a strategy context with $\text{dom}(G) \cap \text{dom}(F) = \emptyset$, and π be an outcome of $G \circ F$ from ℓ .

$$\begin{aligned}
\mathcal{C}, \ell \models_F \varphi &\Leftrightarrow \mathcal{C}, \ell \models_F \langle \cdot A \cdot \rangle \psi \\
&\Leftrightarrow \exists F_A \in \text{Strat}(A). \forall \pi' \in \text{Out}(\ell, F_A \circ F). \mathcal{C}, \pi' \models_{F_A \circ F} \psi \\
&\Leftrightarrow \exists F_A \in \text{Strat}(A). \forall F' \in \text{Strat}(\text{Agt} \setminus (\text{dom}(F) \cup A)). \\
&\hspace{15em} \forall \pi' \in \text{Out}(\ell, F' \circ F_A \circ F). \mathcal{C}, \pi' \models_{F_A \circ F} \psi \\
&\Leftrightarrow \exists F_A \in \text{Strat}(A). \forall F' \in \text{Strat}(\text{Agt} \setminus (\text{dom}(F) \cup A)). \\
&\hspace{15em} \forall \pi' \in \text{Out}(\ell, F' \circ F_A \circ F). \mathcal{C}, \pi' \models_{F' \circ F_A \circ F} \widehat{\psi}^{[\text{dom}(F) \cup A]} \\
&\hspace{15em} \text{(by i.h.)} \\
&\Leftrightarrow \exists F_A \in \text{Strat}(A). \forall F' \in \text{Strat}(\text{Agt} \setminus (\text{dom}(F) \cup A)). \\
&\hspace{15em} \forall \pi' \in \text{Out}(\ell, F' \circ F_A \circ G \circ F). \mathcal{C}, \pi' \models_{F' \circ F_A \circ G \circ F} \widehat{\psi}^{[\text{dom}(F) \cup A]} \\
&\hspace{15em} \text{(because } F' \circ F_A \circ G \circ F = F' \circ F_A \circ F\text{)} \\
&\Leftrightarrow \exists F_A \in \text{Strat}(A). \forall F' \in \text{Strat}(\text{Agt} \setminus (\text{dom}(F) \cup A)). \\
&\hspace{15em} \exists \pi' \in \text{Out}(\ell, F' \circ F_A \circ G \circ F). \mathcal{C}, \pi' \models_{F' \circ F_A \circ G \circ F} \widehat{\psi}^{[\text{dom}(F) \cup A]} \\
&\hspace{15em} \text{(because } |\text{Out}(\ell, F' \circ F_A \circ G \circ F)| = 1\text{)} \\
&\Leftrightarrow \exists F_A \in \text{Strat}(A). \mathcal{C}, \ell \models_{F_A \circ G \circ F} \neg \langle \cdot \text{Agt} \setminus (\text{dom}(F) \cup A) \cdot \rangle \neg \widehat{\psi}^{[\text{dom}(F) \cup A]} \\
&\Leftrightarrow \mathcal{C}, \ell \models_{G \circ F} \langle \cdot A \cdot \rangle \neg \langle \cdot \text{Agt} \setminus (\text{dom}(F) \cup A) \cdot \rangle \neg \widehat{\psi}^{[\text{dom}(F) \cup A]} \\
&\Leftrightarrow \mathcal{C}, \ell \models_{G \circ F} \widehat{\varphi}^{[\text{dom}(F)]}
\end{aligned}$$

- If $\varphi = \cdot \rangle A \langle \psi$: again, we prove the second statement. Let G be a strategy context with $\text{dom}(G) \cap \text{dom}(F) = \emptyset$, and π be an outcome of $G \circ F$ from ℓ . We have:

$$\begin{aligned}
\mathcal{C}, \ell \models_F \varphi &\Leftrightarrow \mathcal{C}, \ell \models_F \cdot \rangle A \langle \psi \\
&\Leftrightarrow \mathcal{C}, \ell \models_{F_A} \psi
\end{aligned}$$

$$\begin{aligned}
&\Leftrightarrow \mathcal{C}, \ell \models_{G \circ F \setminus A} \widehat{\psi}^{[\text{dom}(F) \setminus A]} && \text{(by i.h.)} \\
&\Leftrightarrow \mathcal{C}, \ell \models_{G \circ F \setminus A \circ F \setminus A} \widehat{\psi}^{[\text{dom}(F) \setminus A]} && \text{(by Lemma 4)} \\
&\Leftrightarrow \mathcal{C}, \ell \models_{G \circ F} \widehat{\psi}^{[\text{dom}(F) \setminus A]}
\end{aligned}$$

□

COROLLARY 6. *Let φ be an ATL_{sc}^* state-formula, and A be a subset of a given set Agt of agents. Then φ is equivalent to $\widehat{\varphi}^{[A]}$ under any strategy context F with $\text{dom}(F) = A$ for any CGS based on Agt , i.e. for any CGS \mathcal{C} using Agt as its set of players and for any computation ρ , we have:*

$$\mathcal{C}, \rho \models_F \varphi \quad \text{iff} \quad \mathcal{C}, \rho \models_F \widehat{\varphi}^{[\text{dom}(F)]}.$$

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

THEOREM 7. *Given a set of agents Agt , any ATL_{sc}^* 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 ATL^* state formula φ can be translated into the equivalent ATL_{sc} formula $\widehat{\varphi}^\emptyset$ in polynomial time. Thus we have the following corollary:

COROLLARY 8. *Model-checking ATL_{sc} is 2EXPTIME-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 $(k + 1)\text{EXPTIME}$. The program complexity (i.e., the complexity of model-checking a fixed ATL_{sc} formula) is 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 $QD\mu$ [16], but they were only valid for subclasses of CGSs: strategy logic was only studied on turn-based games, while the algorithm for $QD\mu$ 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 ;
- $l: T \rightarrow \Sigma$ 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 $\text{dir}_T(n) = \{s \in S \mid n \cdot s \in T\}$. The set of *successors of n in T* is $\text{succ}_T(n) = \{n \cdot s \mid s \in \text{dir}_T(n)\}$. We use \mathcal{T}_n to denote the subtree rooted in n . An S -tree is complete if $T = S^*$, i.e., if $\text{dir}_T(n) = S$ 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

$$\text{Path}_{\mathcal{T}} = \{s_0 \cdot s_1 \cdots \in S^\omega \mid \forall i \in \mathbb{N}. s_0 \cdot s_1 \cdots s_i \in T\}.$$

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 $\text{Inf}(l(\pi))$ for the set of letters in Σ that appear infinitely 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 $\text{proj}_{\Sigma_i}(\mathcal{T})$, 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:

$$\text{PBF}(P) \ni \zeta ::= p \mid \zeta \wedge \zeta \mid \zeta \vee \zeta \mid \top \mid \perp$$

where p ranges over P . That a valuation $v: P \rightarrow \{\top, \perp\}$ satisfies a formula in $\text{PBF}(P)$ is defined in the natural way. We abusively say that a subset P' of P satisfies a formula $\varphi \in \text{PBF}(P)$ iff the valuation $\mathbb{1}_{P'}$ (mapping the elements of P' to \top and the elements of $P \setminus P'$ to \perp) 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 $\mathcal{A} = \langle Q, q_0, \tau, \text{Acc} \rangle$ where

- Q is a finite set of states;
- $q_0 \in Q$ is the initial state;
- $\tau: Q \times \Sigma \rightarrow \text{PBF}(S \times Q)$ is the transition function;
- $\text{Acc}: Q^\omega \rightarrow \{\top, \perp\}$ 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 $\mathcal{A} = \langle Q, q_0, \tau, \text{Acc} \rangle$ 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

- $p(\varepsilon) = (\varepsilon, q_0)$

- for each node $e \in E$ with $p(e) = (t, q)$, the set $\text{dir}_E(e) = \{(s_0, q_0), (s_1, q_1), \dots, (s_n, q_n)\} \subseteq S \times Q$ 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, for any infinite path $\pi \in (S \times Q)^\omega$ in Path_E , it holds $\text{Acc}(p_Q(\pi)) = \top$. 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 $\Omega: Q \rightarrow \{0, \dots, k-1\}$, from which Acc is defined as follows: $\text{Acc}(p_Q(\pi)) = \top$ iff $\min\{\Omega(q) \mid q \in \text{Inf}(p_Q(\pi))\}$ 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 $\text{idx}(\mathcal{A})$. Analogously, $\langle S, \Sigma \rangle$ -NPTs are $\langle S, \Sigma \rangle$ -NTAs with parity acceptance conditions.

4.2 Unwinding of a CGS

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

For each location $\ell \in \text{Loc}$, we define $\Sigma(\ell) = \{\ell\} \times \{\text{Lab}(\ell)\} \times \{\text{Edg}(\ell)\}$, and $\Sigma^+(\ell) = \Sigma(\ell) \times (\mathcal{M} \cup \{\perp\})^{\text{Agt}} \times 2^{\{p_o, p_l, p_r\}}$, where \perp 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[†] $\Sigma_{\mathcal{C}} = \bigcup_{\ell \in \text{Loc}} \Sigma(\ell)$, and $\Sigma_{\mathcal{C}}^+ = \bigcup_{\ell \in \text{Loc}} \Sigma^+(\ell)$.

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

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 (deterministic) $\langle \overline{\text{Loc}}, \Sigma_{\mathcal{C}}^+ \rangle$ -APT $\mathcal{A}_{\mathcal{C}, \ell_0} = \langle \overline{\text{Loc}}, \bar{\ell}_0, \tau, \Omega \rangle$ s.t.

- $\overline{\text{Loc}} = \{\bar{\ell} \mid \ell \in \text{Loc}\}$,
- $\bar{\ell}_0$ is the initial state;
- given a state $\bar{\ell} \in \overline{\text{Loc}}$ and a letter $\sigma \in \Sigma_{\mathcal{C}}^+$, we consider several cases:
 - if $\sigma \in \Sigma^+(\ell)$, we let

$$\tau(\bar{\ell}, \sigma) = \bigwedge_{\ell' \in \text{Loc}} (\ell', \bar{\ell}').$$

- otherwise, we let $\tau(\bar{\ell}, \sigma) = \perp$
- Ω constantly equals 0 (hence any valid execution tree is accepting).

[†]Notice that $|\Sigma_{\mathcal{C}}| = |\text{Loc}|$ and $|\Sigma_{\mathcal{C}}^+|$ is linear in the size of the input, as we assume an explicit representation of the Edg function [12].

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 $\text{proj}_{\Sigma_{\mathcal{C}}}^+(\mathcal{T})$ is the unwinding of \mathcal{C} from ℓ_0 .*

PROOF. Assume that \mathcal{T} is accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$. Since $\tau(\bar{\ell}, \sigma) = \perp$ when $\sigma \notin \Sigma^+(\ell)$, and since the initial state of the automaton is $\bar{\ell}_0$, it must be the case that $l(\varepsilon) \in \Sigma^+(\ell_0)$. Also, for each $\ell \in \text{Loc}$, any execution tree must contain the node $(\ell, \bar{\ell})$, labelled with $(\ell, \bar{\ell})$.

By induction, it is easily shown that any node n in any accepting execution tree is such that $n = (\ell_i, \bar{\ell}_i)_i$, and that for each $\ell \in \text{Loc}$, $n \cdot (\ell, \bar{\ell})$ is a successor of n . Hence each node n in \mathcal{T} has $|\text{Loc}|$ successors, and is labelled with $l(n) \in \Sigma^+(\text{last}(\ell_0 \cdot n))$. It follows that $\text{proj}_{\Sigma_{\mathcal{C}}}^+(\mathcal{T})$ is the unwinding of \mathcal{C} from ℓ_0 .

The converse direction is easily proved by explicitly building an accepting execution tree. \square

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 . Such an automaton can easily be built, either directly or by applying Lemma 14.

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 $A \in \text{Agt}$, and for each node $n \in T$, we define $\text{strat}_A^{\mathcal{T}}(\ell_0 \cdot n) = l_{\text{str}}(n)(A) \in \mathcal{M} \cup \{\perp\}$. For $D \subseteq \text{Agt}$, we write $\text{strat}_D^{\mathcal{T}}$ for the set of strategies $(\text{strat}_A^{\mathcal{T}})_{A \in D}$.

As a first step, for each $D \subseteq \text{Agt}$, we build a $\langle \text{Loc}, \Sigma_{\mathcal{C}}^+ \rangle$ -APT $\mathcal{A}_{\text{strat}}(D)$ which will ensure that for all $A \in D$, $\text{strat}_A^{\mathcal{T}}$ is *really* a strategy for player A , i.e., never returns \perp . This automaton has only one state q_0 , with $\tau(q_0, \sigma) = \bigwedge_{\ell \in \text{Loc}} (\ell, q_0)$ provided that $\sigma_{\text{str}}(A) \neq \perp$ for all $A \in D$. Otherwise, $\tau(q_0, \sigma) = \perp$. Finally, $\mathcal{A}_{\text{strat}}$ 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 $D \subseteq \text{Agt}$. 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 $\mathcal{A}_{\text{strat}}(D)$ iff for each player $A \in D$, $\text{strat}_A^{\mathcal{T}}$ never equals \perp .*

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

This is achieved by the following two-state automaton $\mathcal{A}_{\text{out}}(D) = \langle Q, q_{\in}, \tau, \Omega \rangle$:

- $Q = \{q_{\in}, q_{\notin}\}$,
- q_{\in} is the initial state,
- the transition function is defined as follows:

$$\begin{aligned} \tau(q_{\in}, \sigma) &= \bigwedge_{\ell \in \text{Next}(\sigma, D)} (\ell, q_{\in}) \wedge \bigwedge_{\ell \notin \text{Next}(\sigma, D)} (\ell, q_{\notin}) && \text{if } p_o \in \sigma \\ &= \perp && \text{otherwise} \end{aligned}$$

$$\begin{aligned} \tau(q_{\neq}, \sigma) &= \bigwedge_{\ell \in \text{Loc}} (\ell, q_{\neq}) && \text{if } p_o \notin \sigma \\ &= \perp && \text{otherwise} \end{aligned}$$

where

$$\begin{aligned} \text{Next}(\sigma, D) &= \{\ell \in \text{Loc} \mid \exists (m_i)_i \in \mathcal{M}^{\text{Agt}} \text{ s.t.} \\ &\quad (\sigma_{\text{Loc}}, \ell) = \sigma_{\text{Edg}}(\sigma_{\text{Loc}}, (m_i)_i) \text{ and } \forall A_i \in D. \sigma_{\text{str}}(A_i) = m_i.\} \end{aligned}$$

In other terms, $\text{Next}(\sigma, D)$ 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 $\text{Next}(\sigma, D)$ is non-empty iff $\sigma_{\text{str}}(A_i) \neq \perp$ for all $A_i \in D$.

- Again, Ω constantly equals 0, so that any execution tree is accepted.

Automata $\mathcal{A}_{\text{out}}(D)$ have the following property:

LEMMA 13. *Let \mathcal{C} be a CGS, and ℓ_0 be one of its locations, and $D \subseteq \text{Agt}$. 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 $\mathcal{A}_{\text{strat}}(D)$. Then \mathcal{T} is accepted by $\mathcal{A}_{\text{out}}(D)$ iff for any node $n \in T$, it holds $p_o \in l_p(n)$ iff the finite run $\ell_0 \cdot n$ is an outcome of $\text{strat}_D^{\mathcal{T}}$ from ℓ_0 .*

PROOF. Assume \mathcal{T} is accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$, $\mathcal{A}_{\text{strat}}(D)$ and $\mathcal{A}_{\text{out}}(D)$. We prove the equivalence by induction on the depth of the node n . When $n = \varepsilon$, $\ell_0 \cdot n$ is an outcome of any strategy from ℓ_0 . Also, since the initial state is q_{ε} , it must be the case that $p_o \in l(\varepsilon)$, because $\tau(q_{\varepsilon}, \sigma) = \perp$ if $p_o \notin \sigma$.

Now, pick any node $n = m \cdot \ell$ in \mathcal{T} , different from the root, and assume that the equivalence holds for the predecessor node m of n . First, assume that $p_o \in l_p(n)$. Notice that $\mathcal{A}_{\text{out}}(D)$ is a deterministic tree automaton. The execution tree of $\mathcal{A}_{\text{out}}(D)$ on \mathcal{T} is a tree $\mathcal{U} = \langle T, v \rangle$ where $v: T \rightarrow T \times \{q_{\varepsilon}, q_{\neq}\}$. By construction of the transition function τ , since $p_o \in l_p(n)$, it cannot be the case that $v(n) = (n, q_{\neq})$, hence $v(n) = (n, q_{\varepsilon})$. Then if $v(m) = (m, q_{\neq})$, then all successors of this node would be q_{\neq} -nodes. Hence $v(m) = (m, q_{\varepsilon})$. By induction, this entails that the finite run $\ell_0 \cdot m$ is an outcome of $\text{strat}_D^{\mathcal{T}}$ from ℓ_0 . Also, it must be the case that $\ell \in \text{Next}(l(m), D)$, which precisely entails that $\ell_0 \cdot n$ is also an outcome of $\text{strat}_D^{\mathcal{T}}$ from ℓ_0 . Conversely, if $p_o \notin l_p(n)$, then the execution tree must contain the node (n, q_{\neq}) . We distinguish two cases, whether the predecessor node is (m, q_{ε}) or (m, q_{\neq}) . In the former case, we get that $\ell_0 \cdot m$ is an outcome, but that $\ell \notin \text{Next}(l(m), D)$, which means that $\ell_0 \cdot n$ is not an outcome of the strategies for D encoded in \mathcal{T} . In the latter case, $\ell_0 \cdot m$ is already not an outcome, and neither is $\ell_0 \cdot n$.

We now prove the converse implication. Let \mathcal{T} be a $\Sigma_{\mathcal{C}}^+$ -labelled complete LOC-tree accepted by $\mathcal{A}_{\mathcal{C}, \ell_0}$ and $\mathcal{A}_{\text{strat}}(D)$, and such that nodes labelled by p_o precisely describe the outcomes of $\text{strat}_D^{\mathcal{T}}$ from ℓ_0 . Consider the tree $\mathcal{U} = \langle U, v \rangle$ where

$$U = \{(n_i, q_i)_i \mid (n_i)_i \in T \text{ and for all } i, q_i = q_{\varepsilon} \text{ if } p_o \in l_p((n_j)_{j \leq i}) \text{ and } q_i = q_{\neq} \text{ otherwise}\}$$

and $v((n_i, q_i)_{i \leq m}) = ((n_i)_{i \leq m}, q_m)$. We prove that this is a valid execution tree. Indeed, it starts from the initial state q_{ε} , since ℓ_0 is an outcome of any strategy from itself. Now, consider some node $n' = (n_i, q_i)_i$ in U , corresponding to the node $n = (n_i)_i$ of T . If $p_o \in l_p(n)$,

then $v(n')$ is (n, q_\in) , and $\ell_0 \cdot n$ is an outcome of strat_D^T from ℓ_0 . By definition of $\text{Next}(\sigma, D)$, for all $\ell \in \text{Loc}$, it holds $p_o \in l_p(n \cdot \ell)$ iff $\ell \in \text{Next}(l(n), D)$. Then U contains $n' \cdot (\ell, q_\in)$ for all $\ell \in \text{Next}(l(n), D)$, and $n' \cdot (\ell, q_\notin)$ for all $\ell \notin \text{Next}(l(n), D)$. The transition function is satisfied at this node. Now, if $p_o \notin l_p(n)$, then $v(n')$ is (n, q_\notin) , and $\ell_0 \cdot n$ is not an outcome, thus for all $\ell \in \text{Loc}$, $\ell_0 \cdot n \cdot \ell$ is also not an outcome. It immediately follows that U contains all the successors of n' needed to satisfy the transition condition. \square

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. The first three lemmas are classical results, and we only provide a proof for the fourth one.

LEMMA 14.[14, 15] *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 \bar{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 $\max(\text{idx}(\mathcal{A}), \text{idx}(\mathcal{B})) + 1$, while those of \mathcal{D} are $|\mathcal{A}|$ and $\text{idx}(\mathcal{A})$.*

LEMMA 15.[15] *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 $|\mathcal{N}| \in 2^{O(|\mathcal{A}| \text{idx}(\mathcal{A}) \cdot \log(|\mathcal{A}| \text{idx}(\mathcal{A})))}$ and $\text{idx}(\mathcal{N}) \in O(|\mathcal{A}| \text{idx}(\mathcal{A}))$.*

LEMMA 16.[13] *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*

$$\mathcal{T} \in \mathcal{L}(\mathcal{B}_i) \iff \exists \mathcal{T}' \in \mathcal{L}(\mathcal{A}). \text{proj}_{\Sigma_i}(\mathcal{T}) = \text{proj}_{\Sigma_i}(\mathcal{T}').$$

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 $\text{proj}_{\Sigma}(\mathcal{T}) = \text{proj}_{\Sigma}(\mathcal{T}')$, 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*

$$\mathcal{T} \in \mathcal{L}(\mathcal{B}) \iff \forall n \in T. (p \in l(n) \text{ iff } \mathcal{T}_n \in \mathcal{L}(\mathcal{A})).$$

Then \mathcal{B} has size $O(|\mathcal{A}|)$ and index $\text{idx}(\mathcal{A}) + 1$.

PROOF. The construction of \mathcal{B} is in two steps: first, applying Lemma 14, we build an automaton \mathcal{C} accepting exactly the trees in $(\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{P})) \cup (\mathcal{L}(\bar{\mathcal{A}}) \cap \mathcal{L}(\bar{\mathcal{P}}))$, where \mathcal{P} is a one-state automaton accepting exactly the trees whose root is labelled with p . Then \mathcal{C} has size $O(|\mathcal{A}|)$ and index $\text{idx}(\mathcal{A}) + 1$.

Then \mathcal{B} is obtained by forking an execution of \mathcal{C} in all node: formally, writing $\mathcal{C} = \langle Q_C, q_C, \tau_C, \Omega_C \rangle$, then $\mathcal{B} = \langle Q_C \cup \{q_B\}, q_B, \tau_B, \Omega_B \rangle$ (assuming $q_B \notin Q_C$) where τ_B and Ω_B respectively coincide with τ_C and Ω_C on Q_C , and $\tau_B(q_B, \sigma) = \bigwedge_{\ell \in \text{Loc}} (\ell, q_B) \wedge \tau_C(q_C, \sigma)$ and $\Omega_B(q_B) = 0$ (or any even value).

If a $\Sigma \times 2^{\{p\}}$ -labelled S -tree \mathcal{T} is accepted by \mathcal{B} , then for all node $n \in \mathcal{T}$, any execution tree contains a node labelled with (n, q_B) . From such a node, \mathcal{B} forks new branches to q_B -nodes on the one hand, and mimicks the behaviour of \mathcal{C} from q_C on the other hand. The latter implies that the subtree \mathcal{T}_n is accepted by \mathcal{C} , so that we have the required equivalence.

Conversely, let \mathcal{T} be a $\Sigma \times 2^{\{p\}}$ -labelled S -tree satisfying the right-hand-side equivalence. Then for all node n of \mathcal{T} , the subtree \mathcal{T}_n is accepted by \mathcal{C} , and thus admits an accepting execution tree. Based on these execution trees, one easily comes up with an accepting execution tree of \mathcal{B} on \mathcal{T} . \square

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 $D \subseteq \text{Agt}$ be a coalition. We can build a $\langle \text{Loc}, \Sigma_C^+ \rangle$ -APT $\mathcal{A}_{\psi, D}$ s.t.*

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

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

- for any two Σ_C^+ -labelled complete Loc -tree \mathcal{T} and \mathcal{T}' s.t. $\text{proj}_{\Sigma_C'}(\mathcal{T}) = \text{proj}_{\Sigma_C'}(\mathcal{T}')$, with $\Sigma_C' = \Sigma_C \times (\mathcal{M} \cup \{\perp\})^{\text{Agt}}$, 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.

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 turn to the case of ATL_{sc} modalities $\langle A \cdot \rangle \mathbf{X} \varphi$, $\langle A \cdot \rangle \varphi_1 \mathbf{U} \varphi_2$ and $\dagger \langle A \cdot \rangle \varphi_1 \mathbf{R} \varphi_2$. We give a detailed proof of the easier case of $\langle A \cdot \rangle \mathbf{X} \varphi$, and then briefly explain how it can be adapted to handle the other two modalities.

The general idea of the construction is as follows: we use automaton $\mathcal{A}_{\text{out}}(D \cup A)$ 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 , in order to get our new automaton for $\langle A \cdot \rangle \mathbf{X} \varphi$.

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 $\mathcal{B}_{p_r, \varphi, D \cup A}$ that accepts a tree \mathcal{T} iff the nodes labelled by p_r are exactly the roots of subtrees accepted by $\mathcal{A}_{\varphi, D \cup A}$. Notice that acceptance by $\mathcal{A}_{\varphi, D \cup A}$ does not depend on the labelling with p_r , thanks to the second property of the Lemma. Applying the induction hypothesis, we get that, given a tree $\mathcal{T} = \langle T, l \rangle$ accepted by $\mathcal{A}_{\mathcal{C}}$ and $\mathcal{A}_{\text{strat}}(D \cup A)$, it holds

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

\dagger The “release” modality \mathbf{R} is the dual of \mathbf{U} . Notice that \mathbf{X} is self-dual as we only evaluate formulas along infinite outcomes.

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 CTL^{*} property $\mathbf{A}(\mathbf{G} p_o \rightarrow \mathbf{X} p_r)$. We refer to [11] for this classical construction. This automaton \mathcal{A}_f has the following property: for any Σ_C^+ -labelled Loc-tree $\mathcal{T} = \langle T, l \rangle$, we have

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

Now, let \mathcal{H} be the product of $\mathcal{A}_{\text{strat}}(A)$, $\mathcal{A}_{\text{out}}(D \cup A)$, \mathcal{A}_f and $\mathcal{B}_{p_r, \varphi, D \cup A}$, and let \mathcal{T} be a tree accepted by \mathcal{A}_C and $\mathcal{A}_{\text{strat}}(D)$. If \mathcal{T} is accepted by \mathcal{H} , then

- we have $D \cup A \subseteq \text{dom}(\mathcal{T})$, and from Lemma 13, the branches whose nodes are labelled with p_o are exactly the outcomes of $\text{strat}_{D \cup A}^{\mathcal{T}}$ from $l_{\text{Loc}}(\varepsilon)$;
- from (2), those outcomes satisfy $\mathbf{X} p_r$;
- from (1), any node n labelled with p_r corresponds to a state where φ holds under strategy $\text{strat}_{D \cup A}^{\mathcal{T}_n}$.

In other terms, if \mathcal{T} is accepted by \mathcal{H} , then $D \cup A \subseteq \text{dom}(\mathcal{T})$ and all the outcomes of the strategy $\text{strat}_{D \cup A}^{\mathcal{T}}$ from $l_{\text{Loc}}(\varepsilon)$ satisfy $\mathbf{X}\varphi$, which we can write as $\mathcal{C}, l_{\text{Loc}}(\varepsilon) \models \langle \cdot \rangle \mathbf{X}\varphi$.

The converse does not hold in general, but we prove a weaker form: from $\mathcal{T} = \langle T, l \rangle$, accepted by \mathcal{A}_C and $\mathcal{A}_{\text{strat}}(D)$, and such that $D \cup A \subseteq \text{dom}(\mathcal{T})$ and the outcomes of $\text{strat}_{D \cup A}^{\mathcal{T}}$ from $l_{\text{Loc}}(\varepsilon)$ satisfy $\mathbf{X}\varphi$, we build $\mathcal{T}' = \langle T', l' \rangle$ such that $\text{proj}_{\Sigma_C^+}(\mathcal{T}) = \text{proj}_{\Sigma_C^+}(\mathcal{T}')$, 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 $\mathcal{A}_{\text{out}}(D \cup A)$ and $\mathcal{B}_{p_r, \varphi, D \cup A}$. Since we don't modify the "strategy"-part of the labelling, it holds $\text{strat}_{\text{Agt}}^{\mathcal{T}} = \text{strat}_{\text{Agt}'}^{\mathcal{T}'}$ we still have $D \cup A \subseteq \text{dom}(\mathcal{T}')$, and $\mathcal{C}, l_{\text{Loc}}(\varepsilon) \models_{\text{strat}_{D \cup A}^{\mathcal{T}'}} \langle \cdot \rangle \mathbf{X}\varphi$. As a consequence, the outcomes of $\text{strat}_{D \cup A}^{\mathcal{T}'}$ from $l_{\text{Loc}}(\varepsilon)$, which we have labelled with p_o , all satisfy $\mathbf{X}\varphi$, so that their second state is labelled with p_r . It follows 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}_C and $\mathcal{A}_{\text{strat}}(D)$,

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

Now, applying Lemma 15, we get a (Σ_C^+, Loc) -NPT \mathcal{N} such that $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{H})$. We can then apply Lemma 16 for $\Sigma_C^+ = (\Sigma_C \times (\mathcal{M} \cup \{\perp\})^{\text{Agt} \setminus A}) \times ((\mathcal{M} \cup \{\perp\})^A \times 2^{p_o, p_l, p_r})$ on the NPT \mathcal{N} ; the resulting (Σ_C^+, Loc) -NPT \mathcal{P} accepts all trees \mathcal{T} whose labelling on $(\mathcal{M} \cup \{\perp\})^A \times 2^{p_o, p_l, p_r}$ 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}_C and by $\mathcal{A}_{\text{strat}}(D)$. 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 $\Sigma_C \times (\mathcal{M} \cup \{\perp\})^{\text{Agt} \setminus A}$, and accepted by \mathcal{N} . Since $\mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{H})$, and from (3), we get that $D \cup A \subseteq \text{dom}(\mathcal{T}')$ and $\mathcal{C}, l_{\text{Loc}}(\varepsilon) \models_{\text{strat}_{D \cup A}^{\mathcal{T}'}} \langle \cdot \rangle \mathbf{X}\varphi$. Thus $\text{strat}_A^{\mathcal{T}'}$ is a strategy for coalition A , and it witnesses the fact that $\mathcal{C}, l_{\text{Loc}}(\varepsilon) \models_{\text{strat}_D^{\mathcal{T}'}} \langle A \rangle \mathbf{X}\varphi$, and we get the desired result since $\text{strat}_D^{\mathcal{T}} = \text{strat}_D^{\mathcal{T}'}$. Conversely, if $\mathcal{C}, l_{\text{Loc}}(\varepsilon) \models_{\text{strat}_D^{\mathcal{T}}} \langle A \rangle \mathbf{X}\varphi$, then we can modify the labelling of \mathcal{T} with a witnessing strategy for A , obtaining a tree \mathcal{T}' such

that $\mathcal{C}, l_{\text{Loc}}(\varepsilon) \models_{\text{strat}_{D \cup A}^{\mathcal{T}'}} \langle \cdot \rangle \mathbf{X} \varphi$. From (3), \mathcal{T}' can in turn be modified into a tree \mathcal{T}'' , with $\text{proj}_{\Sigma_C'}(\mathcal{T}'') = \text{proj}_{\Sigma_C'}(\mathcal{T}')$, in such a way that $\mathcal{T}'' \in \mathcal{L}(\mathcal{H})$. Finally, since the projections of \mathcal{T}'' and \mathcal{T} coincide on $(\Sigma_C \times (\mathcal{M} \cup \{\perp\})^{\text{Agt} \setminus A})$, it holds that \mathcal{T} is accepted by \mathcal{P} .

The proofs for the “until” and “release” modalities follow the same lines, using p_l and p_r as extra atomic propositions for the left- and right-hand subformulas of these modalities, 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, we handle the case of $\langle \cdot \rangle A \langle \cdot \rangle \varphi$ formulas. For a coalition D , we let $\mathcal{A}_{\langle \cdot \rangle A \langle \cdot \rangle \varphi, D} = \mathcal{A}_{\varphi, D \setminus A}$. From the induction hypothesis, this automaton satisfies the second condition. Now, pick \mathcal{T} accepted by \mathcal{A}_C and $\mathcal{A}_{\text{strat}}(D)$. Then \mathcal{T} is also accepted by $\mathcal{A}_{\text{strat}}(D \setminus A)$ (as a consequence of Lemma 12). Then

$$\begin{aligned} \mathcal{T} \in \mathcal{L}(\mathcal{A}_{\langle \cdot \rangle A \langle \cdot \rangle \varphi, D}) \quad \Leftrightarrow \quad \mathcal{T} \in \mathcal{L}(\mathcal{A}_{\varphi, D \setminus A}) \quad \Leftrightarrow \\ \mathcal{C}, l_\ell(\varepsilon) \models_{\text{strat}_{D \setminus A}^{\mathcal{T}}} \varphi \quad \Leftrightarrow \quad \mathcal{C}, l_\ell(\varepsilon) \models_{\text{strat}_D^{\mathcal{T}}} \langle \cdot \rangle A \langle \cdot \rangle \varphi. \end{aligned}$$

Unless $A = \emptyset$, the construction of the automaton for $\langle \cdot \rangle A \langle \cdot \rangle \mathbf{X} \varphi$ (or $\langle \cdot \rangle A \langle \cdot \rangle \varphi_1 \mathbf{U} \varphi_2$ or $\langle \cdot \rangle A \langle \cdot \rangle \varphi_1 \mathbf{R} \varphi_2$) 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. \square

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.*

$$\mathcal{L}(\mathcal{A}) \neq \emptyset \quad \Leftrightarrow \quad \mathcal{C}, \ell_0 \models_{\emptyset} \varphi.$$

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 $\mathcal{A}_{\varphi, \emptyset}$ (given by Lemma 18) with \mathcal{A}_{C, ℓ_0} . In case this $\langle \text{Loc}, \Sigma_C^+ \rangle$ -APT accepts a tree \mathcal{T} , Lemma 18 entails that $\mathcal{C}, \ell_0 \models_{\emptyset} \varphi$. 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 $l_{\text{str}}(n) = \perp$ for all $n \in T$ is accepted by \mathcal{A}_{C, ℓ_0} (and, trivially, by $\mathcal{A}_{\text{strat}}(\emptyset)$), and from Lemma 18, it is also accepted by $\mathcal{A}_{\varphi, \emptyset}$. \square

PROOF (OF THEOREM 9). The first statement of Theorem 9 directly follows, since emptiness of alternating parity tree automata \mathcal{A} can be checked in time $\exp(O(|\mathcal{A}| \times \text{idx}(\mathcal{A})))$.

For the second statement, notice that the size and index of $\mathcal{A}_{\varphi, \emptyset}$ 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 $\mathcal{A}_{\text{out}}(D)$ requires the computation of $\text{Next}(\sigma, D)$). Non-emptiness is then checked in time exponential in $|\mathcal{C}|$. \square

Remarks: Our algorithm can easily be modified in order to handle ATL_{sc}^* . One solution is to rely on Theorem 7, but remember that our translation from ATL_{sc}^* 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 CTL* formula. This would result 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 $\mathcal{A}_{\text{out}}(D)$, 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 ATL_{sc}^* 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 (extended abstract). Technical Report 08-01, Clausthal University 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. *Journal of the 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. Press, 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, LNCS, p. 59–73. Springer, 2007.
- [9] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Information and Computation*, 208(6):677–693, 2010.

- [10] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronizations skeletons using branching-time temporal logic. In *LOP'81*, LNCS 131, p. 52–71. Springer, 1982.
- [11] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model-checking. *Journal of the ACM*, 47(2):312–360, 2000.
- [12] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4(2), 2008.
- [13] 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.
- [14] D. E. Muller and P. E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.
- [15] 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. *Theoretical Computer Science*, 141(1-2):69–107, 1995.
- [16] S. Pinchinat. A generic constructive solution for concurrent games with expressive constraints on strategies. In *ATVA'07*, LNCS 4762, p. 253–267. Springer, 2007.
- [17] A. Pnueli. The temporal logic of programs. In *FOCS'77*, p. 46–57. IEEE Comp. Soc. Press, 1977.
- [18] J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *SOP'82*, LNCS 137, p. 337–351. Springer-Verlag, 1982.
- [19] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In *TARK'07*, p. 269–278, 2007.