Verifying Agents with Memory is Harder Than It Seemed

Nils Bulling [a], Wojciech Jamroga [b]
[a] *Clausthal University of Technology, Germany*
[b] *University of Luxembourg, Luxembourg*

Abstract. $\mathbf{ATL^+}$ is a variant of alternating-time temporal logic that does not have the expressive power of full $\mathbf{ATL^*}$, but still allows for expressing some natural properties of agents. It has been believed that verification with $\mathbf{ATL^+}$ is $\mathbf{\Delta_3^P}$-complete for both memoryless agents and players who can memorize the whole history of the game. In this paper, we show that the latter result is not correct. That is, we prove that model checking $\mathbf{ATL^+}$ for agents that use strategies with memory is in fact **PSPACE**-complete. On a more optimistic note, we show that fairness constraints can be added to $\mathbf{ATL^+}$ without further increasing the complexity of model checking, which makes $\mathbf{ATL^+}$ an attractive alternative to the full language of $\mathbf{ATL^*}$.

Keywords: Strategic logic, model checking, complexity

1. Introduction

The alternating-time temporal logic $\mathbf{ATL^*}$ and its less expressive version **ATL** [1,3] have been studied extensively in previous years. Much research was focused on the way such logics can be used for the verification of multi-agent systems, *model checking* being the most important method in this respect. Consequently, the computational complexity of model checking turned out essential to evaluate and compare the practical usability of different variants of strategic logics. It is known that the model checking problem of full $\mathbf{ATL^*}$ is **2EXPTIME**-complete, and only **P**-complete for **ATL**, if perfect recall strategies are used [3], i.e., if players can memorize the whole history of the game. Hence, the latter variant of the logic is more attractive computationally. However, there is also a price to pay in terms of expressiveness. The simple property that an agent can make p true infinitely often can, for instance, be expressed in $\mathbf{ATL^*}$ but not in **ATL**.

A tradeoff is offered by $\mathbf{ATL^+}$, a variant of the alternating-time temporal logic that does not have the expressive power of full $\mathbf{ATL^*}$, but still allows for expressing some natural properties of agents. For example, we can use the formula $\langle\langle robot\rangle\rangle(\Diamond\mathsf{cleanRoom} \wedge \Diamond\mathsf{packageDelivered})$ to demand that the robot can clean the room and deliver the package, without specifying in which order the tasks should be accomplished.

Verification with $\mathbf{ATL^+}$ has been believed to be $\mathbf{\Delta_3^P}$-complete for both memoryless and perfect-recall strategies [17]. In this paper, we show that the latter result is wrong. That is, we prove that model checking $\mathbf{ATL^+}$ for agents that use strategies with full memory is in fact **PSPACE**-complete. Since the $\mathbf{\Delta_3^P}$-completeness for the memoryless semantics still holds, we get that memory makes verification harder already for $\mathbf{ATL^+}$, and not just for $\mathbf{ATL^*}$ as it was believed. We also show that fairness conditions can be added to $\mathbf{ATL^+}$ without further increasing the complexity.

The rest of this paper is structured as follows. In Section 2 we introduce the relevant logics and their models, and discuss the variant $\mathbf{ATL^+}$ in more detail. In Section 3 we correct the existing "results" on the model checking complexity of $\mathbf{ATL^+}$ for agents with perfect information and perfect recall. In Section 4 we study $\mathbf{EATL^+}$ (i.e., $\mathbf{ATL^+}$ augmented with the temporal operator $\overset{\infty}{\Diamond}$ for "infinitely often"). Finally, we argue why we consider the results significant and present some conclusions in Sections 5 and 6, respectively.

This article is an extended and revised version of the conference papers [5,6].

2. $\mathbf{ATL^+}$ and the Matter of Recall

We begin by introducing the strategic logics that will be discussed in this paper. The *alternating-time temporal logic* $\mathbf{ATL^*}$ [1,3] is a temporal logic that incorporates some basic game-theoretical notions. Essentially, $\mathbf{ATL}^*$ generalizes the branching time logic $\mathbf{CTL}^*$ [7] by replacing the path quantifiers E, A with *cooperation modalities* $\langle\!\langle A \rangle\!\rangle$. Informally, $\langle\!\langle A \rangle\!\rangle\gamma$ expresses that the group of agents A has a collective strategy to enforce temporal property γ. $\mathbf{ATL}^*$ formulae include temporal operators: $\bigcirc$ ("in the next state") and $\mathcal{U}$ ("until"). Additional operators $\Diamond$ ("now or sometime in the future") and $\Box$ ("always from now on") can be defined as $\Diamond\gamma \equiv \top\,\mathcal{U}\,\gamma$ and $\Box\gamma \equiv \neg\Diamond\neg\gamma$. It should be noted that the path quantifiers A, E of $\mathbf{CTL}^*$ can be expressed in $\mathbf{ATL}^*$ with $\langle\!\langle \emptyset \rangle\!\rangle$, $\langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle$ respectively.

2.1. Syntax and Variants

In the rest of the paper we assume that Π is a nonempty set of *proposition symbols* and $\mathbb{A}\mathrm{gt}$ a nonempty and finite set of *agents*. Alternating-time temporal logic comes in several variants, of which $\mathbf{ATL}^*$ is the broadest. Formally, the language of $\mathbf{ATL}^*$ is given by formulae φ generated by the grammar below, where $A \subseteq \mathbb{A}\mathrm{gt}$ is a set of agents, and $\mathsf{p} \in \Pi$ is an atomic proposition:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle\gamma,$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc\gamma \mid \gamma\,\mathcal{U}\,\gamma.$$

Formulae φ are called *state formulae*, and γ *path formulae* of $\mathbf{ATL}^*$.

The best known variant of the alternating time temporal logics is $\mathbf{ATL}$ (sometimes called "$\mathbf{ATL}$ without star" or "vanilla" $\mathbf{ATL}$) in which every occurrence of a cooperation modality is immediately followed by exactly one temporal operator.[1] In this paper, however, we study the model checking problem for $\mathbf{ATL}^+$, a variant that sits between $\mathbf{ATL}^*$ and $\mathbf{ATL}$. The language of $\mathbf{ATL}^+$ includes only formulae where each temporal operator is followed by a state formula, and allows cooperation modalities to be followed by *a Boolean combination* of path subformulae; i.e. path formulae are defined by $\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi$.

Formally, $\mathbf{ATL}^+$ formulae are defined by the following grammar:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle\gamma,$$
$$\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi.$$

[1]In that case, the language is augmented explicitly with the "always" operator $\Box$ [1,3] or, more generally, the "weak until" operator $\mathcal{W}$ [12], because otherwise maintenance/safety properties cannot be expressed.

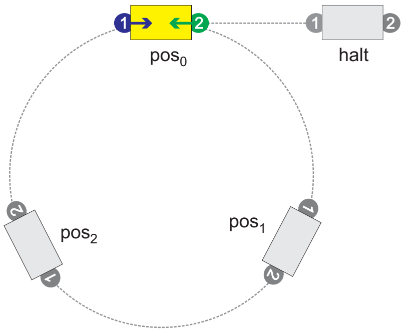


Fig. 1. Two robots and a carriage: a schematic view

Example 1 *The* $\mathbf{ATL}$ *formula* $\langle\!\langle jamesbond \rangle\!\rangle\Diamond\mathsf{win}$ *says that James Bond can eventually win, no matter how the other agents act. On the other hand,* $\langle\!\langle jamesbond \rangle\!\rangle\Box(\mathsf{assigned} \rightarrow \Diamond\mathsf{accomplished})$ *is an* $\mathbf{ATL}^*$ *formula which clearly belongs to neither* $\mathbf{ATL}$ *nor* $\mathbf{ATL}^+$ *and deems agent 007 to be able to accomplish all his future missions. Finally,* $\langle\!\langle jamesbond \rangle\!\rangle(\Box\neg\mathsf{crash} \wedge \Diamond\mathsf{land})$ *(James Bond can prevent the space ship from crashing and make it eventually land) is a formula of* $\mathbf{ATL}^+$ *but not of* $\mathbf{ATL}$.

2.2. Semantics

The semantics of $\mathbf{ATL}^*$ is defined over a variant of transition systems where transitions are labeled with combinations of actions, one per agent. Formally, a *concurrent game structure* ($\mathbf{CGS}$) is a tuple $M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, Act, d, o \rangle$ which includes a nonempty finite set of all agents $\mathbb{A}\mathrm{gt} = \{1, \dots, k\}$, a nonempty set of states St, a set of atomic propositions Π and their valuation $\pi : \Pi \rightarrow 2^{St}$, and a nonempty finite set of (atomic) actions Act. Function $d : \mathbb{A}\mathrm{gt} \times St \rightarrow 2^{Act}$ defines nonempty sets of actions available to agents at each state, and o is a (deterministic) transition function that assigns the outcome state $q' = o(q, \alpha_1, \dots, \alpha_k)$ to state q and a tuple of actions $\langle \alpha_1, \dots, \alpha_k \rangle$ for $\alpha_i \in d(i, q)$ and $1 \leq i \leq k$, that can be executed by $\mathbb{A}\mathrm{gt}$ in q. Thus, we assume that all the agents execute their actions synchronously; the combination of the actions, together with the current state, determines the next transition of the system.

In the rest of the paper, we will write $d_i(q)$ instead of $d(i, q)$, and we will denote the set of collective choice of group A at state q by $d_A(q) = \prod_{i \in A} d_i(q)$. A *path* $\lambda = q_0 q_1 q_2 \dots$ is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}. We use

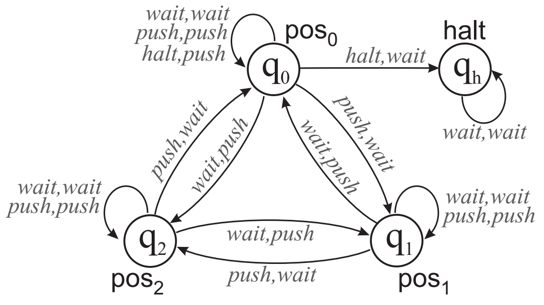


Fig. 2. Two robots and a carriage: concurrent game structure M_1 that models the scenario

$\lambda[i]$ to denote the ith position on path λ (starting from $i = 0$) and $\lambda[i, \infty]$ to denote the subpath of λ starting from i.

Example 2 (Robots and Carriage) *Consider the scenario depicted in Figures 1 and 2. Two robots push a carriage from opposite sides. As a result, the carriage can move clockwise or anticlockwise, or it can remain in the same place. We assume that each robot can either push (action* push*) or refrain from pushing (action* wait*). Moreover, they both use the same force when pushing. Thus, if the robots push simultaneously or wait simultaneously, the carriage does not move. When only one of the robots is pushing, the carriage moves accordingly. Finally, when the carriage is in position* 0*, robot* 1 *may try to retire it to a halting position. The halting is successful if the other robot is not pushing at the same time.*

To make our model of the domain discrete, we identify 4 *different positions of the carriage, and associate them with states* q_0*,* q_1*,* q_2*, and* q_h*. We label the states with propositions* $\mathsf{pos}_0, \mathsf{pos}_1, \mathsf{pos}_2, \mathsf{halt}$*, respectively, to allow for referring to the current position of the carriage in the object language.*

A *strategy* of agent a is a plan that specifies what a is going to do in each situation. It makes sense, from a conceptual and computational point of view, to distinguish between two types of strategies: an agent may base his decision on the current state or on the whole history of events that have happened. Also, the agent may have complete or incomplete knowledge about the current global state of the system throughout the game. To distinguish between those cases, we use the taxonomy and notation introduced in [17]: $\mathbf{ATL_{xy}}$ where $x = i$ (resp. *I*) stands for *imperfect* (resp. *perfect*) *information* and $y = r$ (resp. *R*) for *imperfect* (resp. *perfect*) *recall*. Here we are mainly interested in the perfect-information+prefect-recall setting.

A *perfect-information-perfect-recall strategy* (*IR-strategy*) for agent a is a function $s_a : St^+ \to Act$ such that $s_a(q_0q_1 \dots q_n) \in d_a(q_n)$ for any finite history $q_0q_1 \dots q_n$. A *perfect-information-memoryless strategy* (*Ir-strategy*) is a function $s_a : St \to Act$ such that $s_a(q) \in d_a(q)$ for each q. We do not consider the model checking problem for imperfect information games in this paper, so we will omit definitions of *ir*- and *iR*-strategies here.

A *collective strategy* for a group of agents $A = \{a_1, \dots, a_r\} \subseteq \mathbb{A}\mathrm{gt}$ is simply a tuple of individual strategies $s_A = \langle s_{a_1}, \dots, s_{a_r} \rangle$. By $s_A|_a$, we denote agent a's part s_a of the collective strategy s_A, for $a \in A$. Function $out(q, s_A)$ returns the set of all paths that may occur when agents A execute strategy s_A from state q onward. For an *IR*-strategy we have:

$out(q, s_A) = \{\lambda = q_0q_1q_2 \dots \mid q_0 = q$ and for each $i = 1, 2, \dots$ there exists a tuple of agents' decisions $\langle \alpha^{i-1}_{a_1}, \dots, \alpha^{i-1}_{a_k} \rangle$ such that $\alpha^{i-1}_a \in d_a(q_{i-1})$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha^{i-1}_a = s_A|_a(q_0q_1 \dots q_{i-1})$ for every $a \in A$, and $o(q_{i-1}, \alpha^{i-1}_{a_1}, \dots, \alpha^{i-1}_{a_k}) = q_i\}$.

The definition for memoryless strategies is analogous.

Let M be a $\mathbf{CGS}$, q a state, and λ a path in M. The semantics of $\mathbf{ATL^*_{xy}}$ is defined as follows [3,17]:

- $M, q \models \mathsf{p}$ iff $q \in \pi(\mathsf{p})$, for $\mathsf{p} \in \Pi$;
- $M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;
- $M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;
- $M, q \models \langle\langle A \rangle\rangle\gamma$ iff there is an xy-strategy s_A for agents A such that for each path $\lambda \in out(s_A, q)$ we have $M, \lambda \models \gamma$.
- $M, \lambda \models \varphi$ iff $M, \lambda[0] \models \varphi$;
- $M, \lambda \models \neg\gamma$ iff $M, \lambda \not\models \gamma$;
- $M, \lambda \models \gamma_1 \wedge \gamma_2$ iff $M, \lambda \models \gamma_1$ and $M, \lambda \models \gamma_2$;
- $M, \lambda \models \bigcirc\gamma$ iff $M, \lambda[1, \infty] \models \gamma$; and
- $M, \lambda \models \gamma_1 \mathcal{U} \gamma_2$ iff there is an $i \in \mathbb{N}_0$ such that $M, \lambda[i, \infty] \models \gamma_2$ and $M, \lambda[j, \infty] \models \gamma_1$ for all $0 \leq j < i$.

Example 3 (Robots and Carriage, ctd.) *Since the outcome of each robot's action depends on the current action of the other robot, no agent can make sure that the carriage moves to any particular position. So, we have for example that* $M_1, q_0 \models \neg\langle\langle 1 \rangle\rangle\Diamond\mathsf{pos}_1$*. On the other hand, the robots can cooperate to move the carriage. For instance, it holds that* $M_1, q_0 \models \langle\langle 1, 2 \rangle\rangle\Diamond\mathsf{pos}_1$ *(example strategy: robot* 1 *always pushes and robot* 2 *always waits).*

In fact, the same strategy can be used to express that the robots can make the carriage visit every "active" *position, which is captured by the following* $\mathbf{ATL}^+$ *satisfaction:* $M_1, q_0 \models \langle\!\langle 1,2 \rangle\!\rangle(\Diamond \mathsf{pos}_0 \wedge \Diamond \mathsf{pos}_1 \wedge \Diamond \mathsf{pos}_2)$*. Note that all the above properties hold for both memoryless agents and agents with perfect recall.*

2.3. Importance of ATL$^+$

It is well known that the memoryless and perfect-recall semantics for $\mathbf{ATL}$ formulae coincide [3,17]. That is, $M, q \models \varphi$ in $\mathbf{ATL_{IR}}$ iff it holds in $\mathbf{ATL_{Ir}}$. The same is *not* true for $\mathbf{ATL}^*$, and in fact, already for $\mathbf{ATL}^+$. For example, formula $\langle\!\langle 1,2 \rangle\!\rangle(\Diamond \mathsf{pos}_1 \wedge \Diamond \mathsf{halt})$ holds in M_1, q_0 in the set of perfect-recall strategies but not in the set of memoryless strategies. In consequence, $\mathbf{ATL}^+$ can be seen as the minimal well-known syntactic variant of the alternating-time logic that distinguishes between the memoryless and perfect-recall semantics.

Moreover, note that the *IR*- and *Ir*-semantics yield different validities for $\mathbf{ATL}^+$. For example, formula $\langle\!\langle A \rangle\!\rangle(\Diamond \mathsf{p}_1 \wedge \Diamond \mathsf{p}_2) \leftrightarrow \langle\!\langle A \rangle\!\rangle \Diamond \big((\mathsf{p}_1 \wedge \langle\!\langle A \rangle\!\rangle \Diamond \mathsf{p}_2) \vee (\mathsf{p}_2 \wedge \langle\!\langle A \rangle\!\rangle \Diamond \mathsf{p}_1) \big)$ is valid in the perfect-recall semantics (*IR*), but not in the memoryless variant (*Ir*). Since "vanilla" $\mathbf{ATL_{IR}}$ and $\mathbf{ATL_{Ir}}$ have the same validities, $\mathbf{ATL}^+$ can be also seen as the minimal variant of the alternating time logics for which the *IR*- and *Ir*-semantics give rise to different *logics* in the traditional sense (as sets of valid sentences).

In conceptual terms, we can use $\mathbf{ATL}^+$ to specify a set of goals that should be achieved without saying in which order they should be accomplished, like in the formula $\langle\!\langle robot \rangle\!\rangle(\Diamond \mathsf{cleanRoom} \wedge \Diamond \mathsf{packageDelivered})$. Moreover, $\mathbf{ATL}^+$ allows for reasoning about what can be achieved under certain assumptions about the agents' behavior, as Example 4 shows. This kind of properties has been especially studied in deontic logic and normative systems (e.g., [14,15,20,18]), but also in reasoning about plausible behavior of agents [4].

Example 4 *Consider a class of systems, each represented by a concurrent game structure* M *and a collection of* behavioral constraint sets $\mathcal{B}_a$*, one per agent* $a \in \mathbb{A}\mathrm{gt}$*. Like in [18], we take every behavioral constraint from* $\mathcal{B}_a$ *to be a pair* (q, α)*, with the underlying interpretation that action* α *is forbidden for agent* a *in state* q *(for instance, by a social norm or law). Such representations can be reconstructed into a single* $\mathbf{CGS}$ M' *by adding special propositions* V_a*,* $a \in \mathbb{A}\mathrm{gt}$*, with the intuitive meaning "agent* a *has committed a violation with its last action". If necessary, several copies of an original state* q *from* M *can be created, with different configurations of the* V_a *labels. Note that* M' *is only linearly larger than* M *wrt the number of original transitions in* M*.*

Now, property "a *can enforce property* γ *while complying with social norms" can be captured in* M' *by the* $\mathbf{ATL}^+$ *formula* $\langle\!\langle a \rangle\!\rangle((\Box \neg \mathsf{V}_a) \wedge \gamma)$*. A similar property, "*b *can enforce* γ *provided that* a *complies with norms" can be expressed with* $\langle\!\langle b \rangle\!\rangle((\Box \neg \mathsf{V}_a) \rightarrow \gamma)$*.*

2.4. Expressivity and Complexity of Model Checking

$\mathbf{ATL}^+$ is more expressive than $\mathbf{ATL}$, which follows from the fact that the "weak until" operator is expressible in $\mathbf{ATL}^+$ (as $\varphi \mathcal{W} \psi \equiv \varphi \mathcal{U} \psi \vee \Box(\neg\psi)$), but not in the original version of $\mathbf{ATL}$ [12]. Still, many formulae of $\mathbf{ATL}^+$ have their equivalent counterparts in $\mathbf{ATL}$. For instance, the $\mathbf{ATL}^+$ formula $\langle\!\langle jamesbond \rangle\!\rangle(\Box \neg \mathsf{crash} \wedge \Diamond \mathsf{land})$ from Example 1 can be equivalently rephrased in $\mathbf{ATL}$ as $\langle\!\langle jamesbond \rangle\!\rangle(\neg \mathsf{crash})\, \mathcal{U}\, (\mathsf{land} \wedge \langle\!\langle jamesbond \rangle\!\rangle \Box \neg \mathsf{crash})$.

In particular, we have that $\mathbf{ATL}^+_{\mathbf{IR}}$ formulae can be equivalently translated into $\mathbf{ATL_{IR}}$ *with the "weak until" operator* [10]. We observe that in some cases the translation results in an exponential blowup of the length of the formula. Thus, $\mathbf{ATL}^+_{\mathbf{IR}}$ has the same expressive power as "vanilla" $\mathbf{ATL_{IR}}$ with "weak until", but it seems to allow for exponentially more succinct and intuitive specifications of some properties (in a similar way to $\mathbf{CTL}^+$ vs. $\mathbf{CTL}$, cf. [19]).

Regarding the complexity of verification for strategic logics, the following patterns can be observed (albeit not without exceptions):

- Model checking more expressive logics is usually harder than the less expressive ones (example: $\mathbf{ATL}^*$ vs. $\mathbf{ATL}$);
- Model checking more succinct logics is usually harder than the less succinct ones (example: $\mathbf{ATL}^+$ vs. $\mathbf{ATL}$ with "weak until");
- Model checking more expressive models is usually harder than the less expressive ones (example: imperfect information vs. perfect information agents);
- Model checking more succinct models is usually harder than the less succinct ones (example: agents with perfect recall vs. memoryless agents).

Indeed, for the memoryless semantics (*Ir*), model checking of $\mathbf{ATL_{Ir}}$ can be done in linear time wrt the number of transitions in the model and the length of the formula [3],[2] while model checking of $\mathbf{ATL}^+_{\mathbf{Ir}}$ is $\mathbf{\Delta^P_3}$-complete,[3] and model checking of $\mathbf{ATL}^*_{\mathbf{Ir}}$ is $\mathbf{PSPACE}$-complete. Moreover, for the perfect recall semantics (*IR*), model checking of $\mathbf{ATL_{IR}}$ is still linear (it is *the same* logic after all) while verification of $\mathbf{ATL}^*_{\mathbf{IR}}$ is complete in double exponential time [3].

What about model checking $\mathbf{ATL}^+_{\mathbf{IR}}$? In [17], it is claimed to be $\mathbf{\Delta^P_3}$-complete, so apparently no price is paid for assuming agents' memory in this case. Unfortunately, the claim is wrong. We show in Section 3 that the problem becomes $\mathbf{PSPACE}$-complete in the *IR*-setting. Note that the results in [12] on the verification of $\mathbf{ATL}^+_{\mathbf{IR}}$ in various non-standard settings are also incorrect since they crucially depend on the claim from [17]; we will correct them in Section 3.3.

3. Model Checking $\mathbf{ATL}^+_{IR}$

In an excellent study [17], Schobbens claims that model checking $\mathbf{ATL}^+$ is $\mathbf{\Delta^P_3}$-complete wrt to the number of transitions in the model and the length of the formula, for both perfect-recall and memoryless semantics. For memoryless agents, the upper bound can be shown by the following algorithm. Given a formula $\langle\!\langle A\rangle\!\rangle\gamma$ with no nested cooperation modalities, we can guess a memoryless strategy of A, "trim" the model accordingly, model-check the $\mathbf{CTL}^+$ formula $\mathsf{E}\neg\gamma$ in the resulting model, and revert the result. Note that a memoryless strategy can be guessed in polynomially many steps, and the trimming process requires only polynomially many steps too. For nested cooperation modalities, we repeat the procedure recursively (bottom-up). Since model checking of the $\mathbf{CTL}^+$ formula $\mathsf{E}\neg\gamma$ can be done in nondeterministic polynomial time [13], we get that the overall procedure runs in time $\left(\mathbf{P^{NP}}\right)^{\mathbf{NP}} = \mathbf{P}^{\left(\mathbf{NP^{NP}}\right)} = \mathbf{\Delta^P_3}$ [17].

For agents with perfect recall, a similar argument *seems* correct. Every formula of $\mathbf{ATL}^+_{\mathbf{IR}}$ can be translated to an equivalent formula of $\mathbf{ATL_{IR}}$ with weak until [10], and for $\mathbf{ATL}$ (also with weak until) it does not make a difference whether the perfect-recall or memoryless semantics is used, so memoryless strategies can be used instead. Hence, it is enough to guess a memoryless strategy, trim the model etc. Unfortunately, this line of reasoning is wrong because the result of the translation (the $\mathbf{ATL_{IR}}$ formula) may include *exponentially many* cooperation modalities (instead of one in the original $\mathbf{ATL}^+_{\mathbf{IR}}$ formula). For example, formula $\langle\!\langle A\rangle\!\rangle(\Diamond \mathsf{p}_1 \wedge \Diamond \mathsf{p}_2)$ is translated to $\langle\!\langle A\rangle\!\rangle\Diamond\big((\mathsf{p}_1\wedge\langle\!\langle A\rangle\!\rangle\Diamond\mathsf{p}_2)\vee(\mathsf{p}_2\wedge\langle\!\langle A\rangle\!\rangle\Diamond\mathsf{p}_1)\big)$; for a longer list of achievement goals $(\Diamond\mathsf{p}_1 \wedge \cdots \wedge \Diamond\mathsf{p}_n)$, every permutation must be explicitly enumerated. Thus, we may need to guess exponentially many polynomial-size strategies, which clearly cannot be done in polynomial time.

There seems to be an intuitive way of recovering from the problem. Note that, in an actual execution, only a polynomial number of these strategies will be used. So, we can try to first guess a sequence of goals (in the right order) for which strategies will be needed, then the strategies themselves, fix those strategies in the model (cloning the model into as many copies as we need) and check the corresponding $\mathbf{CTL}^+$ formula in it. Unfortunately, this is also wrong: for different execution paths, we may need *different* ordering of the goals (and hence strategies). And we have to consider exponentially many paths in the worst case.

So, what is the complexity of model checking $\mathbf{ATL}^+_{\mathbf{IR}}$ in the end? The problem turns out to be harder than $\mathbf{\Delta^P_3}$, namely $\mathbf{PSPACE}$-complete.

3.1. Lower Bound

We prove the $\mathbf{PSPACE}$-hardness by a reduction of Quantified Boolean Satisfiability (QSAT), a canonical $\mathbf{PSPACE}$-complete problem.

Definition 1 (QSAT [16])
Input: *A Boolean formula Φ in negation normal form (i.e., negations occur only at literals) with n propositional variables $x_1,\dots,x_n$.*
Output: *True if $\exists x_1\forall x_2\dots Q_n x_n\ \Phi$ holds, and false otherwise (where $Q_n=\forall$ if n is even, and $Q_n=\exists$ if n is odd).*

Given an instance of QSAT we construct a turn-based[4] concurrent game structure M with two players: the *verifier* $\mathbf{v}$ and the *refuter* $\mathbf{r}$. The structure consists of the following sections:

[2] It is important to add that the "weak until" operator $\mathcal{W}$ does not increase the complexity [12].

[3] $\mathbf{\Delta^P_3} = \mathbf{P}^{\mathbf{\Sigma^P_2}}$ is the class of problems that can be solved by a deterministic Turing machine that makes adaptive queries to an oracle of type $\mathbf{\Sigma^P_2} = \mathbf{NP^{NP}}$. That is, the oracle is a nondeterministic TM that can query another oracle (a nondeterministic TM itself). All the three machines are required to run in polynomial time.

[4] A model is *turn-based* if for each state there is a single agent that controls the subsequent transition, and the other agents have no real choice there (which can be modeled by assuming $d_q(a)=\{wait\}$ for every agent a except the "owner" of q).

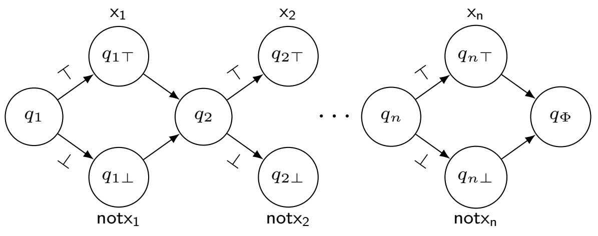


Fig. 3. Construction of the concurrent game structure for QSAT: value choice section

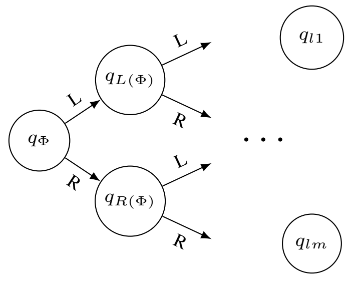


Fig. 4. $\mathbf{CGS}$ for QSAT: formula structure section

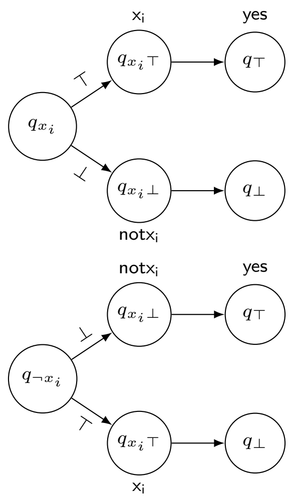


Fig. 5. $\mathbf{CGS}$ for QSAT: sections of literals

– *Value choice section:* a sequence of states q_i, one per variable x_i, where the values of x_i's will be "declared", see Figure 3. States q_i with odd i are controlled by $\mathbf{v}$, states with even i are controlled by $\mathbf{r}$. The owner of a state can choose between two possible valuations ($\top, \perp$). Choosing $\top$ leads to a state where proposition $\mathsf{x_i}$ holds; choosing $\perp$ leads to a state labeled by proposition $\mathsf{notx_i}$.
– *Formula structure section:* corresponds to the parse tree of Φ, see Figure 4. For every subformula Ψ of Φ, there is a state q_Ψ with two choices: L leading to state $q_{L(\Psi)}$ and R leading to $q_{R(\Psi)}$, where $L(\Psi)$ is the left hand side subformula of Ψ and $R(\Psi)$ is the right hand side subformula of Ψ. The verifier controls q_Ψ if the outermost connective in Ψ is a disjunction; the refuter controls the state if it is a conjunction. Note that each leaf state in the tree is named according to a literal l_i from Φ, that is, either with a variable x_i or its negation $\neg x_i$.
– *Sections of literals:* for every literal l in Φ, we have a single state q_l, controlled by the owner of the Boolean variable x_i in l. Like in the value choice section, the agent chooses a value ($\top$ or $\perp$) for the variable (not for the literal!) which leads to a new state labeled with the proposition $\mathsf{x_i}$ (for action $\top$) or $\mathsf{notx_i}$ (for $\perp$). Finally, the system proceeds to the winning state $q_\top$ (labeled with the proposition yes) if the valuation of x_i makes the literal l true, and to the losing state $q_\perp$ otherwise – see Figure 5 for details, and Figure 6 for an example of the whole construction.
Note that, if the values of variables x_i are assigned uniformly at states q_l (that is, the actions executed at q_l form a valuation of $x_1, \dots, x_n$), then the formula structure section together with the sections of literals implement the game theoretical semantics [11] of formula Φ given the valuation.

Note that the value of variable x_i can be declared twice during an execution of the model(first in the value choice section, and then in the section of literals). The following "consistency" macro: $\mathsf{Cons}_i \equiv \Box\neg\mathsf{x_i} \vee \Box\neg\mathsf{notx_i}$ expresses that the value of x_i cannot be declared both $\top$ and $\perp$ during a single execution. Now, for the *IR*-semantics, we have that:

Lemma 1 $\exists x_1 \forall x_2 \dots Q_n x_n \ \Phi$ *iff*

$$M, q_1 \models \langle\langle \mathbf{v} \rangle\rangle \Big(\bigwedge_{i \in Odd} \mathsf{Cons}_i \wedge \big(\bigwedge_{i \in Even} \mathsf{Cons}_i \rightarrow \Diamond \mathsf{yes} \big) \Big).$$

Proof. The informal idea is as follows. The $\mathbf{ATL}^+$ formula specifies that $\mathbf{v}$ can consistently assign values to "his" variables, so that if $\mathbf{r}$ consistently assigns values to "his" variables (in any way), formula Φ will always evaluate to $\top$, which is exactly the meaning of QSAT. The way a player assigns a value to variable x_i may depend on what has been assigned to $x_1, \dots, x_{i-1}$. Note

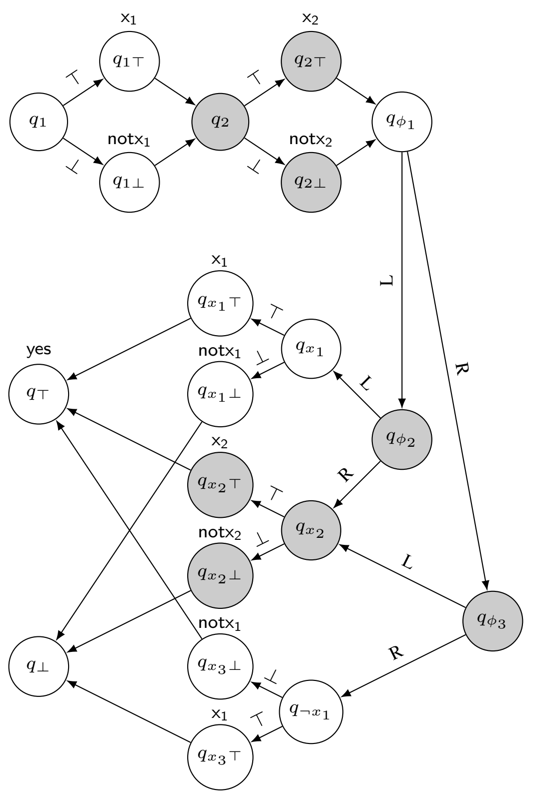


Fig. 6. Concurrent game structure for the QSAT instance $\exists x_1 \forall x_2 (x_1 \wedge x_2) \vee (x_2 \wedge \neg x_1)$. The following shorthands are used in the formula structure section: $\phi_1 \equiv (x_1 \wedge x_2) \vee (x_2 \wedge \neg x_1), \phi_2 \equiv x_1 \wedge x_2, \phi_3 \equiv x_2 \wedge \neg x_1$. "White" states are owned by the verifier; "grey" states are owned by the refuter.

that this is the reason why perfect recall is necessary to obtain the reduction.

The statement can be formally proved in the following way. We consider wlog only QSAT instances with even alternations of quantifiers (the case for odd n is done analogously). Firstly, note that a QSAT instance $\varphi = \exists x_1 \forall x_2 \dots \forall x_n \Phi(x_1, \dots, x_n)$ evaluates to true iff there is a (partial) function $f : \{\top, \bot\}^* \to \{\top, \bot\}$ such that for all valuations v_i of all x_i with $i \leq n$ and i even the following formula is valid:

$$\Phi(f(\epsilon), v_2, f(v_1 v_2), v_3, \dots, f(v_1 \dots v_{n-1}), v_n)$$

where $v_1 := f(\epsilon)$, $v_3 := f(v_1 v_2)$,..., $v_{n-1} := f(v_1 \dots v_{n-2})$. It is easy to see that if such a function exists then it provides a satisfying valuation for the existential quantifiers in the QSAT instance. Conversely, non-existence of such a function contradicts the existence of such a valuation. We say that f *witnesses* φ. Given a word $w = w_1 \dots w_m \in \{\top, \bot\}^m$ of length $m \geq 2i$, we use $f_{2i+1}(w)$ to denote the value $f(w_1 \dots w_{2i})$ and $f_{2i}(w)$ to denote w_{2i}. Intuitively, $f_i(w)$ returns the assignment v_i of x_i given the choices made before.

"⇒": Let f be a witness for φ. We define the following strategy $s_{\mathbf{v}}$ for $\mathbf{v}$ for histories h of the form $q_1 q^1 q_2 q^2 \dots$ of length at most $2n$ and each $q^i \in \{q_{i\top}, q_{i\bot}\}$. For states in which only one action is applicable we suppose that is it chosen by default. That is, h is a finite path through the value choice section of length at most $2n$. Moreover, we define mapping $\delta : St^* \to \{\top, \bot\}^*$ to map each sequence of states to a word over $\{\top, \bot\}$ as follows: $\delta(\epsilon) = \epsilon$, $\delta(q_{i\top}) = \top$, $\delta(q_{i\bot}) = \bot$, and $\delta(q) = \epsilon$ for all other states; finally, $\delta(qh) = \delta(q)\delta(h)$. Then, we define

$$s_{\mathbf{v}}(hq) := \begin{cases} f(\delta(h)) & \text{for } q = q_{2i+1} \\ nop & \text{for } q \in \{q_{2i+1\top}, q_{2i+1\bot}\} \end{cases}$$

In each subformula state q_ψ "owned" by $\mathbf{v}$, the verifier chooses action L (resp. R) if $L(\psi)$ (resp. $R(\psi)$) evaluates to true given f and h (we write $\psi(f, h) = \top$ for $\psi(f_1(\delta(h)), \dots, f_n(\delta(h)))$ is true):

$$s_{\mathbf{v}}(hq_\psi) := \begin{cases} L & \text{if } L(\psi)(f, h) = \top \\ R & \text{else} \end{cases}$$

Analogously, the action for literal states is chosen

$$s_{\mathbf{v}}(hq_l) := \begin{cases} \top & \text{if } l = x_i, f_i(\delta(h)) = \top) \vee \\ & \quad (l = \neg x_i, f_i(\delta(h)) = \bot)) \\ \bot & \text{else.} \end{cases}$$

It remains to show that $s_{\mathbf{v}}$ is a winning strategy of $\mathbf{v}$. Firstly, it is easily seen that $\bigwedge_{i \in Odd} \mathsf{Cons}_i$ holds for any path of the outcome $out(q_1, s_{\mathbf{v}})$; assuming the contrary contradicts the definition of f. Finally, assuming that there is a counter-strategy of the refuter such that state $q_\bot$ is reached and $\bigwedge_{i \in Even} \mathsf{Cons}_i$ also contradicts that f witnesses φ (this can be shown by structural induction on Φ).

"⇐": Let $s_{\mathbf{v}}$ be a winning strategy for $\mathbf{v}$. We define function f in the obvious way, i.e., $f(\epsilon) := s_{\mathbf{v}}(q_1)$, $f(f(\epsilon)\top) := s_{\mathbf{v}}(q_1 q_{(1 f(\epsilon))} q_2 q_{2\top} q_3)$, $f(f(\epsilon)\bot) := s_{\mathbf{v}}(q_1 q_{(1 f(\epsilon))} q_2 q_{2\bot} q_3)$, etc. We prove that f is a witness for φ.

First, let us observe that for every path in $out(q_1, s_{\mathbf{v}})$, it must hold that $\Box \neg \mathsf{x_i} \vee \Box \neg \mathsf{notx_i}$. In consequence, $s_{\mathbf{v}}(q_1 \dots q_{x_i}) = s_{\mathbf{v}}(q_1 \dots q_{\neg x_i}) = s_{\mathbf{v}}(q_1 \dots q_i)$ for all histories leading to a literal state $q_l \in \{q_{x_i}, q_{\neg x_i}\}$.[5] That is, the formula structure section and the sections of literals define the game semantics of Φ with the valuation of $x_1, \dots, x_n$ given uniformly by the f above. Moreover, every path in $out(q_1, s_{\mathbf{v}})$ must lead to a state where yes holds (i.e., to $q_\top$), which means that Φ evaluates to $\top$ given f, and thus f is a witness for φ.■

We observe that the construction results in a model with $O(|\Phi|)$ states and transitions, and it can be constructed in $O(|\Phi|)$ steps, so we get the following result where the size of a $\mathbf{CGS}$ is defined as the number of its transitions (m) plus the number of states (n). Note, that $\mathcal{O}(n+m) = \mathcal{O}(m)$.

Theorem 2 *Model checking $\mathbf{ATL}^+$ with the perfect-recall semantics is $\mathbf{PSPACE}$-hard with respect to the size of the model and the length of the formula. It is $\mathbf{PSPACE}$-hard even for turn-based models with two agents and "flat" $\mathbf{ATL}^+$ formulae, i.e., ones that include no nested cooperation modalities.*

3.2. Upper Bound

In this section we show that model checking $\mathbf{ATL}^+_{\mathbf{IR}}$ can be done in polynomial space. Our proof has been inspired by the construction in [13], proposed for $\mathbf{CTL}^+$. We begin by introducing some notation.

We say that s_A is a *strategy for* (M, q, γ) if for all $\lambda \in out_M(q, s_A)$ it holds that $M, \lambda \models \gamma$. An $\mathbf{ATL}^+$-path formula γ is called *atomic* if it has the form $\bigcirc \varphi_1$ or $\varphi_1 \,\mathcal{U}\, \varphi_2$ where $\varphi_1, \varphi_2 \in \mathbf{ATL}^+$. For $\varphi \in \mathbf{ATL}^+$ we denote the set of all atomic path subformulae of φ by $\mathcal{APF}(\varphi)$. And, as before, we call an $\mathbf{ATL}^+$-path formula γ *flat* if it does not contain any more cooperation modalities.

Now we can define the notion of *witness position* which is a specific position on a path that "makes" a path formula true or false.

Definition 2 (Witness position) *Let γ be a flat atomic path formula, and let λ be a path. The* witness position *$witpos(\lambda, \gamma)$ of γ wrt λ is defined as follows:*
(1) if $\gamma = \bigcirc \varphi$ then $witpos(\lambda, \gamma) = 1$;
(2) if $\gamma = \varphi_1 \,\mathcal{U}\, \varphi_2$ and

- *$\lambda \models \gamma$ then $witpos(\lambda, \gamma) = \min\{i \geq 0 \mid \lambda[i] \models \varphi_2\}$*
- *$\lambda \not\models \gamma$ and $\lambda \models \Diamond \varphi_2$ then $witpos(\lambda, \gamma) = \min\{i \geq 0 \mid \lambda[i] \models \neg\varphi_1\}$*
- *$\lambda \not\models \gamma$ and $\lambda \not\models \Diamond \varphi_2$ then $witpos(\lambda, \gamma) = -1$.*

Moreover, for a flat (not necessarily atomic) $\mathbf{ATL}^+$ path formula γ, we define the set of witness positions of γ wrt λ as $wit(\lambda, \gamma) = (\bigcup_{\gamma' \in \mathcal{APF}(\gamma)} \{witpos(\lambda, \gamma')\}) \cap \mathbb{N}_0$.

For instance, if formula $\Box\neg\mathsf{p}$ is true on λ then $witpos(\lambda, \Box\neg\mathsf{p}) = -1$ since the formula is an abbreviation for $\neg(\top \,\mathcal{U}\, \mathsf{p})$, and for this formula we have that $witpos(\lambda, \Box\neg\mathsf{p}) = -1$ and consequently, $wit(\lambda, \Box\neg\mathsf{p}) = \emptyset$. In the following we assume that γ is flat.

In the next lemma we show that if there is a strategy that enforces a (flat) path formula γ then the witnesses of all atomic subformulae of γ can be found in a bounded initial fragment of each resulting path. Firstly, we introduce the notion of a *segment* which can be seen as a "minimal loop".

Definition 3 (Segment) *A* segment *of path λ is a tuple $(i, j) \in \mathbb{N}_0^2$ with $i < j$ such that $\lambda[i] = \lambda[j]$ and there are no indices k, k' with $i \leq k < k' \leq j$ such that $\lambda[k] = \lambda[k']$ except for $k = i, k' = j$. The set of segments of λ is denoted by $seg(\lambda)$.*

Lemma 3 *Let $M, q \models \langle\!\langle A \rangle\!\rangle \gamma$. Then, there is a strategy s_A for (M, q, γ) such that for all paths $\lambda \in out_M(q, s_A)$ the following property holds: For every segment $(i, j) \in seg(\lambda)$ with $j \leq \max wit(\lambda, \gamma)$ there is a witness position $k \in wit(\lambda, \gamma)$ with $i \leq k \leq j$.*

Proof. Suppose such a strategy does not exist; then, for any strategy s_A for (M, q, γ), there is a path $\lambda \in out(q, s_A)$ and a segment $(i, j) \in seg(\lambda)$ with $j \leq \max wit(\lambda, \gamma)$ s.t. there is no $k \in wit(\lambda, \gamma)$ with $i \leq k \leq j$.

We now define s'_A as the strategy that is equal to s_A except that it cuts out the "idle" segment (i, j) from λ, i.e., $s'_A(\lambda[0, i]h) := s_A(\lambda[0, j]h)$ for all $h \in St^+$, and $s'_A(h) := s_A(h)$ otherwise. Note that $out(q, s'_A) = out(q, s_A)$ except for paths that begin with $\lambda[0, j]$: these are replaced with paths that achieve the remaining witness positions in $j - i$ less steps. Let $[h]_{q, s_A}$ denote the set of all paths λ' such that $h\lambda' \in out(q, s_A)$ where $h \in St^+$. Now it is easy to see that for all $\lambda' \in [\lambda[0, j]]_{q, s_A}$ we have that the path $\lambda[0, j]\lambda'$ satisfies γ if, and only if, the path $\lambda[0, i]\lambda'$ does. Hence, we have that all paths in $out(q, s'_A)$ satisfy γ. Moreover, the latter set of outcomes is non-empty iff $out(q, s_A)$

[5]Technically, that is true only for the literal states reachable in $out(q_1, s_{\mathbf{v}})$, but since the unreachable states are irrelevant, we can fix $s_{\mathbf{v}}(q_1 \dots q_l) := s_{\mathbf{v}}(q_1 \dots q_i)$ for unreachable q_l's too.

is not. By following this procedure recursively, we obtain a strategy that reaches a witness in every segment of each λ up to $\max wit(\lambda,\gamma)$. ■

Given, for instance, an $\mathbf{ATL}^+$ formula $\langle\!\langle A\rangle\!\rangle(\Diamond \mathsf{p} \wedge \Diamond \mathsf{r})$ the previous lemma says that if A has any winning strategy than it also has one such that only the first two segments on each path in the outcome are important to witness the truth of $\Diamond \mathsf{p} \wedge \Diamond \mathsf{r}$. In the next definition we make this intuition formal and define the truth of $\mathbf{ATL}^+$ path formulae on finite initial sequences of states.

Definition 4 ($\models^k$) *Let M be a* $\mathbf{CGS}$*, λ be path in M, and $k \in \mathbb{N}$. The semantics $\models^k$ is defined as follows:*

$M,\lambda \models^k \neg\gamma$ iff $M,\lambda \not\models^k \gamma$;
$M,\lambda \models^k \gamma \wedge \delta$ iff $M,\lambda \models^k \gamma$ and $M,\lambda \models^k \delta$;
$M,\lambda \models^k \bigcirc\varphi$ iff $M,\lambda[1] \models \varphi$ and $k > 1$; and
$M,\lambda \models^k \varphi\,\mathcal{U}\,\psi$ iff there is an $i < k$ such that $M,\lambda[i] \models \psi$ and $M,\lambda[j] \models \varphi$ for all $0 \le j < i$;

Essentially, we consider the first k states on a path in order to see whether a formula is made true on it.

We define the notion *k-witness positions* on a finite segment of length k in an obvious way: if the witness of γ on the full path λ is $> k$ then the k-witness is -1; otherwise, it is equal to $wit(\lambda,\gamma)$.

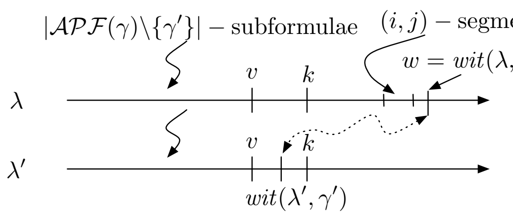
Definition 5 (k-witness strategy) *We say that a strategy s_A is a k-*witness *strategy for (M,q,γ) if for all $\lambda \in out(q,s_A)$ we have that $M,\lambda \models^k \gamma$.*

The following theorem is essential for our model checking algorithm. The result ensures that the existence of a winning strategy can be decided by only guessing the first k-steps of a k-witness strategy.

Theorem 4 *$M,q \models \langle\!\langle A\rangle\!\rangle\gamma$ iff there is a $|St_M| \cdot |\mathcal{APF}(\gamma)|$-witness strategy for (M,q,γ).*

Proof. "$\Rightarrow$:" Let s_A be a strategy for (M,q,γ). By Lemma 3 and the fact that $|wit(\lambda,\gamma)| \le |\mathcal{APF}(\gamma)|$ for any path λ there is a strategy s'_A for (M,q,γ) such that $\max wit(\lambda,\gamma) \le |St_M| \cdot |\mathcal{APF}(\gamma)|$ for all $\lambda \in out(q,s_A)$. This shows that s'_A is a $|St_M| \cdot |\mathcal{APF}(\gamma)|$-witness strategy for (M,q,γ).

"$\Leftarrow$": Suppose there is a $k := |St_M| \cdot |\mathcal{APF}(\gamma)|$ witness strategy then there also is a k-witness strategy such that on no path in the outcome there is an "idle" segment (i,j) (a segment containing no witness) with $j \le v$, where v is the maximal witness on the path smaller than k (cf. Lemma 3). We call such strategies *efficient*. Now suppose there is an efficient k-witness strategy s_A but no strategy for (M,q,γ); i.e. for all efficient k-witness strategies there is a path $\lambda \in out(q,s_A)$ such that $M,\lambda \not\models \gamma$. Note, that this can only happen if there is some $\gamma' \in \mathcal{APF}(\gamma)$ with (minimal) $w := witpos(\lambda,\gamma') \ge k$ that cannot be prevented by A (cf. Figure 7). Due to efficiency all subformulae that have a witness $\le k$ actually have a witness $\le k - |St_M|$. But then, the opponents can ensure that there is some other path $\lambda' \in out(q,s_A)$ on which γ' is witnessed within the first k steps on λ' and after all the other formulae with a witness $\le v$ (i.e. within steps v and k). This contradicts that s_A is a k-witness strategy.



$|\mathcal{APF}(\gamma)\backslash\{\gamma'\}|$ – subformulae
(i,j) – segment
$w = wit(\lambda,\gamma')$
λ
λ'
v
k
v
k
$wit(\lambda',\gamma')$

Fig. 7. Proof of Theorem 4

The see that the opponents can ensure that γ' is witnessed within the first k steps, consider a segment (i,j) such that $j \le w$ and j maximal. In particular, the proponents cannot prevent the sequence $\lambda[j,w]$. But then, the opponents are able to execute there moves played from $\lambda[j]$ onwards already from $\lambda[i]$. This results in a path λ'' also belonging to the outcome which equals λ but segment (i,j) beeing cut out. Following this procedure recursively shows that there is a path λ' such that γ' is witnessed within the first k steps as stated above. ■

In the next theorem we construct an alternating Turing machine that solves the model checking problem.

Theorem 5 *Let $\varphi \equiv \langle\!\langle A\rangle\!\rangle\gamma$ be a flat $\mathbf{ATL}^+_{IR}$ formula, M a* $\mathbf{CGS}$*, and q a state. Then, there is a polynomial-time alternating Turing machine with $\mathcal{O}(nl)$ alternations (wrt the size of the model and length of the formula) that returns "yes" if $M,q \models \varphi$, and "no" otherwise (where l is the length of φ, k is the number of agents, and n the number of states in M).*

Proof. The idea behind the algorithm can be summarized as follows: coalition A acts as a collective "verifier", and the rest of the agents plays the role of a collective "refuter" of the formula. We first transform γ to its negation normal form.[6] Next, we allow the verifier to nondeterministically construct A's strategy step by step for the first $|St_M| \cdot |\mathcal{APF}(\gamma)|$ rounds ($|\mathbb{A}\mathrm{gt}|$ steps each), while the refuter guesses the most damaging responses of $\mathbb{A}\mathrm{gt} \setminus A$. This gives us a finite path h (of length $|St_M| \cdot |\mathcal{APF}(\gamma)|$) that is the outcome of the best strategy of A against the worst course of events. Then, we implement the game-theoretical semantics of propositional logic [11] as a game between the verifier (who controls disjunction) and the refuter (controlling conjunction). The game reduces the truth value of γ to a (possibly negated) atomic subformula γ_0. Finally, we check if $h \models^{|St_M| \cdot |\mathcal{APF}(\gamma)|} \gamma_0$, and return the answer. The correctness of the construction follows from Theorem 4. ■

For model checking arbitrary $\mathbf{ATL}^+$ formulae, we observe that nested cooperation modalities can be model-checked recursively (bottom-up) in the same way as e.g. in the standard model checking algorithm for $\mathbf{ATL}$ [3]. Since $P^{\mathbf{PSPACE}} = \mathbf{PSPACE}$, we obtain the following as immediate corollary.

Theorem 6 *Model checking* $\mathbf{ATL}^+$ *over* $\mathbf{CGS}$*'s with the perfect-recall semantics is* $\mathbf{PSPACE}$*-complete wrt the size of the model and the length of the formula. It is* $\mathbf{PSPACE}$*-complete even for turn-based models with two agents and "flat"* $\mathbf{ATL}^+$ *formulae.*

3.3. Correcting Related Results

Concurrent game structures specify transitions through a function that defines state transformations for *every combination of simultaneous actions* from $\mathbb{A}\mathrm{gt}$. In other words, transitions are given through an array that defines the outcome state for every combination of a state with k actions available at that state. This is clearly a disadvantage from the computational point of view, since the array is in general exponential with respect to the number of agents: more precisely, we have that $m = O(nd^k)$, where m is the number of (labeled) transitions in the model, n is the number of states, d is the maximal number of choices per state, and k is the number of agents.

Two variants of game structures overcome this problem. In *alternating transition systems* ($\mathbf{ATS}$), used as models in the initial semantics of $\mathbf{ATL}$ [1,2], agents' choices are state transformations themselves rather than abstract labels. In *implicit concurrent game structures* [12], the transition array is defined by Boolean expressions. $\mathbf{ATS}$ and implicit $\mathbf{CGS}$ do not hide exponential blowup in a parameter of the model checking problem (m), and hence the complexity of model checking for these representations is perhaps more meaningful than the results obtained for "standard" $\mathbf{CGS}$. In [12], Laroussinie et al. claim that model checking $\mathbf{ATL}^+_{\mathbf{IR}}$ against $\mathbf{ATS}$ as well as implicit $\mathbf{CGS}$ is $\mathbf{\Delta}^{\mathbf{P}}_{\mathbf{3}}$-complete. Since the proofs are actually based on the flawed result from [17], both claims are worth a closer look. We will briefly summarize both kinds of structures and give correct complexity results in this section.

Alternating Transition Systems. An $\mathbf{ATS}$ is a tuple $M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, \delta \rangle$, where $\mathbb{A}\mathrm{gt}, St, \Pi, \pi$ are like in a $\mathbf{CGS}$, and $\delta : St \times \mathbb{A}\mathrm{gt} \to 2^{2^{St}}$ is a function that maps each pair $(state, agent)$ to a non-empty family of choices with respect to possible next states. The idea is that, at state q, agent a chooses a set $Q_a \in \delta(q, a)$ thus forcing the outcome state to be from Q_a. The resulting transition leads to a state which is in the intersection of all Q_a for $a \in \mathbb{A}\mathrm{gt}$. Since the system is required to be deterministic (given the state and the agents' decisions), $Q_{a_1} \cap ... \cap Q_{a_k}$ must always be a singleton.

Implicit $\mathbf{CGS}$. An implicit $\mathbf{CGS}$ is a concurrent game structure where, in *each* state q, the outgoing transitions are defined by a finite sequence

$$((\varphi_1, q_1), ..., (\varphi_n, q_n)).$$

In the sequence, every q_i is a state, and each φ_i is a Boolean combination of propositions $\hat{\alpha}^a$, where $\alpha \in d(a, q)$; $\hat{\alpha}^a$ stands for "agent a chooses action α". The transition function is now defined as: $o(q, \alpha_1, ..., \alpha_k) = q_i$ *iff i is the lowest index such that* $\{\hat{\alpha_1}^1, ..., \hat{\alpha_k}^k\} \models \varphi_i$. It is required that $\varphi_n \equiv \top$, so that no deadlock can occur. The *size* of an implicit model is given by the number of states, agents, and the length of the sum of the sizes of the Boolean formulae.

Model Checking $\mathbf{ATL}^+$ Is PSPACE-Complete Again. Contrary to [12, Section 3.4.1], where model checking $\mathbf{ATL}^+$ with respect to both $\mathbf{ATS}$ and implicit $\mathbf{CGS}$ is claimed to be $\mathbf{\Delta}^{\mathbf{P}}_{\mathbf{3}}$-complete, we establish the complexity as $\mathbf{PSPACE}$.

Theorem 7 *Model checking* $\mathbf{ATL}^+$ *for* $\mathbf{ATS}$ *and implicit* $\mathbf{CGS}$ *using the perfect-recall semantics is*

[6]I.e., so that negation occurs only in front of atomic path subformulae.

$\mathbf{PSPACE}$*-complete wrt the size of the model and the length of the formula (even for turn-based models with two agents and "flat"* $\mathbf{ATL}^+$ *formulae).*

Proof (sketch). Lower bound. We observe that the number of transitions in a turn-based $\mathbf{CGS}$ is linear in the number of states (n), agents (k), and actions (d). Moreover, each turn-based $\mathbf{CGS}$ has an isomorphic $\mathbf{ATS}$, and an isomorphic implicit $\mathbf{CGS}$; the transformation takes $O(nd)$ steps. This, together with the reduction from Section 3.1, gives us $\mathbf{PSPACE}$-hardness wrt n, k, d and the length of the formula (l) and the encoded transition function for model checking $\mathbf{ATL}^+$ against $\mathbf{ATS}$ as well as implicit $\mathbf{CGS}$.

Upper bound. A close inspection of the algorithm from Section 3.2 reveals that it can easily applied to $\mathbf{ATS}$ and implicit $\mathbf{CGS}$. In each step when a transition is taken one has to evaluate a sequence of Boolean formulae. This can be done in polynomial time wrt to a, k, and the length of the encoding. ■

4. Adding Fairness to ATL$^+$

Fairness conditions allow to focus on computations where no agent is neglected wrt given resources (e.g., access to power supply, processor time, etc.). Fairness is extremely important in asynchronous composition of agents. In general, it may happen that requests of a group $A \subset \mathbb{A}\mathrm{gt}$ are postponed forever in favor of actions from other agents. In consequence, if we want to state any positive property about what A can achieve, we need to refer explicitly to paths where A's actions are always eventually executed. To this end, it is enough to augment $\mathbf{ATL}^+$ with the "always eventually" combination $\Box\Diamond$ as an additional primitive $\overset{\infty}{\Diamond}$.

4.1. EATL$^+$

"Extended $\mathbf{ATL}^+$" ($\mathbf{EATL}^+$) is a subset of $\mathbf{ATL}^*$ obtained by adding another kind of path formulae to $\mathbf{ATL}^+$:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\gamma, \text{ where}$$
$$\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \overset{\infty}{\Diamond}\varphi.$$

Note that in $\mathbf{ATL}^*$ the formula $\overset{\infty}{\Diamond}\varphi$ can be written as $\Box\Diamond\varphi$. Hence, we can also use the ordinary $\mathbf{ATL}^*$ semantics to give truth to $\mathbf{EATL}^+$ formulae.

By the fact that $\mathbf{CTL}^+$ is more expressive than $\mathbf{CTL}$ (which follows from $\mathbf{ECTL}$ being more expressive than $\mathbf{CTL}$ [9]), we conjecture that $\mathbf{EATL}^+$ is also more expressive than $\mathbf{ATL}^+$ (a formal proof is beyond the scope of this article). In particular, we conjecture that fairness constraints cannot be expressed in $\mathbf{ATL}^+$. Hence the importance of $\mathbf{EATL}^+$ which allows for reasoning about the outcome of fair computations in model M. This is extremely important in specification and verification of agents that act in an asynchronous environment. For example, most agent (and multi-agent) programming frameworks assume an asynchronous execution platform. In such settings, the following property from [8] holds.

Proposition 8 ([8]) *Let M be a multi-agent program model, q a state in M, and φ an* $\mathbf{ATL}$ *formula. If there is a path in M, q on which φ never holds, then there must be an agent i in M so that, for every coalition $A \subseteq \mathbb{A}\mathrm{gt} \setminus \{i\}$, we have $M \not\models \langle\langle A\rangle\rangle\Diamond\varphi$.*

In other words, if the design of the program does not guarantee that φ must eventually happen, then the execution platform (agent i in the proposition above) can stall actions of every coalition of "real" agents (from $\mathbb{A}\mathrm{gt} \setminus \{i\}$) and prevent them from achieving φ.

In $\mathbf{EATL}^+$, this can be overcome by putting fairness constraints explicitly in the formula. To make our discussion more concrete, let us assume that M is an asynchronous $\mathbf{CGS}$ as defined in [3]. That is, M is a $\mathbf{CGS}$ where agent k is designated as the *scheduler*. The scheduler's task is to choose the agent whose action is going to be executed, i.e., $d_k(q) = \mathbb{A}\mathrm{gt} \setminus \{k\}$ for every $q \in St$, and for every pair of action profiles $\vec{\alpha}, \vec{\alpha}'$ that agree on the action of agent j we have $o(q, \vec{\alpha}, j) = o(q, \vec{\alpha}', j)$. In our construction, we also assume that transitions by different agents lead to different states ($o(q, \vec{\alpha}, i) \neq o(q, \vec{\alpha}', j)$ for $i \neq j$). Moreover, each state is labeled by proposition act_i, where i is the agent whose action was executed last.

Now, for example, the $\mathbf{EATL}^+$ formula $\langle\langle 1, 2\rangle\rangle((\bigwedge_{i=1}^{k-1} \overset{\infty}{\Diamond}\mathsf{act}_i) \rightarrow \Diamond\mathsf{cleanRoom})$ says that agents 1 and 2 can cooperate to make the room clean *for every course of events on which no agent is stalled forever.*

4.2. Model Checking EATL$^+_{IR}$

In this section we extend the construction from Section 3.2 to obtain an algorithm for $\mathbf{EATL}^+$ under the perfect-recall semantics. Firstly, we define the *set of witnesses* $wit^\infty(\lambda, \gamma)$ for a flat atomic formula $\gamma \equiv \overset{\infty}{\Diamond}\varphi$. If $\lambda \not\models \overset{\infty}{\Diamond}\varphi$ then $wit^\infty(\lambda, \gamma) = \emptyset$; and if $\lambda \models \overset{\infty}{\Diamond}\varphi$

then $wit^{\infty}(\lambda, \gamma) = \{i \mid \lambda[i] \models \varphi\}$. Note that the set is either infinite or empty.

Moreover, an $\mathbf{EATL}^{+}$ path formula γ is called $\overset{\infty}{\Diamond}$-*atomic* if it has the form $\overset{\infty}{\Diamond}\varphi_1$. For $\varphi \in \mathbf{EATL}^{+}$ we denote the set of all $\overset{\infty}{\Diamond}$-atomic flat path subformulae of φ by $\mathcal{APF}^{\infty}(\varphi)$.

In the following we generalize the definition of a segment.

Definition 6 (γ-segment, strict) *A* γ-segment *on a path λ is a tuple $(i, j) \in \mathbb{N}_0^2$ with $i < j$ such that $\lambda[i] = \lambda[j]$ and for each $\gamma' \in \mathcal{APF}^{\infty}(\gamma)$ with $wit^{\infty}(\lambda, \gamma') \neq \emptyset$ there is a witness $w \in wit^{\infty}(\lambda, \gamma')$ such that $i \leq w \leq j$.*

We call a γ-segment (i, j) strict *if there is no other γ-segment (k, l) in it.*

The next proposition shows that such γ-segments always exist on paths on which some $\overset{\infty}{\Diamond}$-atomic flat formula is true. The following proofs are done similarly to the ones given in Section 3.2.

Proposition 9 *Let s_A be a strategy for (M, q, γ). Then, for all paths $\lambda \in out(q, s_A)$ and $t \in \mathbb{N}$ there is a strict γ-segment (i, j) on λ with $i \geq t$.*

Proof. Suppose there is a path in the outcome that does not contain such a γ-segment. Then, as the set of states is finite there must be some position $l \geq t$ on λ such that $\lambda[l, \infty]$ does not contain a witness for some $\gamma' \in \mathcal{APF}^{\infty}(\gamma)$ with $wit(\lambda, \gamma') \neq \emptyset$. But this contradicts $wit(\lambda, \gamma') \neq \emptyset$. If there is no $\overset{\infty}{\Diamond}$-formula true on a path the condition is trivially true. ■

Lemma 10 *Let $M, q \models \langle\langle A \rangle\rangle\gamma$. Then, there is a strategy s_A for (M, q, γ) such that any strict γ-segment (i, j) that contains no more witnesses for any formula from $\mathcal{APF}(\gamma)$ contains at most $|St_M| \cdot |\mathcal{APF}^{\infty}(\gamma)|$ states.*

Proof. We proceed similar to Lemma 3 to make all eventualities from $\mathcal{APF}(\gamma)$ true. Then, we modify the strategy to a strategy s'_A such that any segment (i_l, j_l) contained in any strict γ-segment (i, j) contains some witness of $wit^{\infty}(\lambda, \gamma')$ for each $\gamma' \in \mathcal{APF}^{\infty}(\gamma)$ for that the witness set is non-empty on λ (and which does not contain any more witnesses from formulae from $\mathcal{APF}(\gamma)$). Now, we consider the last segment, say (i_l, j_l), contained in (i, j) (i.e. $j_l = j$). If all formulae $\gamma' \in \mathcal{APF}^{\infty}(\gamma)$ with $wit^{\infty}(\lambda, \gamma') \neq \emptyset$ that have a witness in (i_l, j_l) do also have a witness inside (i, j) but outside (i_l, j_l) then we modify s'_A such that (i_l, j_l) is "removed" from the γ-segment (i, j) by applying the reduction of Lemma 3. If not, we chose the segment next to the last one and so on. The resulting γ-segment (i, j') is $j_l - i_l + 1$ states shorter than (i, j). Applied recursively, this procedure results in a γ-segment that contains at most $|\mathcal{APF}^{\infty}(\gamma)|$ necessary segments which are interconnected by a minimal number of states that do not contain unnecessary segments. The number of states of each segment plus the number of intermediate states between two segments is at most $|St_M|$. Hence, the γ-segment contains at most $|St_M|(|\mathcal{APF}^{\infty}(\gamma)|)$ states. ■

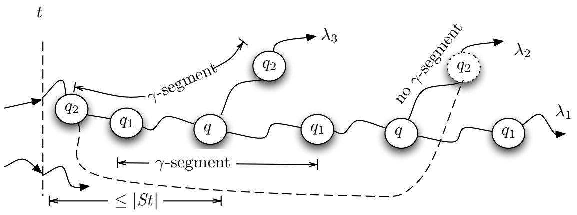


Fig. 8. Proof idea of Theorem 11.

In the following we extend the finite path semantics such that it can deal with $\overset{\infty}{\Diamond}$-atomic flat formulae.

Definition 7 ($\models^k$ for $\mathbf{EATL}^{+}$) *The semantics from Definition 4 is extended to $\mathbf{EATL}^{+}$-formulae by adding the following clause: $M, \lambda \models^k \overset{\infty}{\Diamond}\gamma$ iff there is some $i < k$ such that $M, \lambda[i, \infty] \models^k \gamma$;*

The notion of a k-witness strategy is given analogously to Definition 5: s_A is a *k-witness* strategy for (M, q, γ) if for all $\lambda \in out(q, s_A)$ we have that $M, \lambda \models^k \gamma$.

The analog of Theorem 4 for $\mathbf{EATL}^{+}$ is given next.

Theorem 11 *We have that $M, q \models \langle\langle A \rangle\rangle\gamma$ iff there is a $|St_M| \cdot (1 + |\mathcal{APF}(\gamma)| + |\mathcal{APF}^{\infty}(\gamma)|)$-witness strategy for (M, q, γ).*

Proof (sketch). "$\Rightarrow$": Let s_A be a strategy for (M, q, γ). Then, we modify s_A according to Lemma 3 and obtain a strategy s'_A such that on all paths λ of the outcome of s'_A and for all formulae $\gamma' \in \mathcal{APF}(\gamma)$ with a witness on λ we have that $wit(\lambda, \gamma') \leq |St| \cdot |\mathcal{APF}(\gamma)| =: t$. We modify s'_A to a strategy s''_A according to Proposition 9 and Lemma 10. Finally, the states between t and the start of the strict γ-segment can be shrunk up to at most $|St|$-many, again according to Lemma 3 (cf. Figure 8) resulting in a $|St_M| \cdot (1 + |\mathcal{APF}(\gamma)| + |\mathcal{APF}^{\infty}(\gamma)|)$-witness strategy for (M, q, γ).

"$\Leftarrow$": Now assume there is a $k := |St_M| \cdot (1 + |\mathcal{APF}(\gamma)| + |\mathcal{APF}^\infty(\gamma)|)$-witness strategy for (M, q, γ) and no strategy for (M, q, γ). If this is caused by a formula from $\mathcal{APF}(\gamma)$, or γ' from $\mathcal{APF}^\infty(\gamma)$ with a minimal witness position $\geq k$ the reasoning is as in the proof of Theorem 4. We now consider the case if it is caused by a formula from $\mathcal{APF}^\infty(\gamma)$ with a minimal witness position $< k$. Then, for any k-strategy s_A there must be a $\gamma' \in \mathcal{APF}^\infty(\gamma)$ such that for some path $\lambda_1 \in out(q, s_A)$ it holds that $M, \lambda_1 \models^k \gamma'$ but $M, \lambda_2 \not\models \gamma'$ where λ_2 equals λ_1 up to position k. We show that this cannot be the case. $M, \lambda_1 \models^k \gamma'$ implies that γ' has a witness in the initial γ-segment on λ_1 (cf. the initial γ-segment on λ_1 with start and end state q_1 in Figure 8). So, there must be a state q and an outgoing path λ_2 containing no more γ-segments. However, this state and outgoing path must also be present in the initial γ-segment on the path λ_1 and on λ_3 (see Fig. 8) there must also be a γ-segment. If it starts within q_1 and q on λ_3 it must also be present on λ_2. So, suppose the initial γ-segment on λ_3 with start and end state q_2 begins before q_1. But this gives us a (non-strict) γ-segment on λ_2 (shown by the dotted line) and of course, this segment can also be reached on the outgoing path λ_2 going through state q on λ_1. Applying this reasoning recursively proves that each of these paths contains infinitely many γ-segments. This contradicts the assumption that $M, \lambda_2 \not\models \gamma'$. ■

The previous result allows to construct an alternating Turing machine with a fixed number of alternations to solve the model checking problem (cf. the proof of Theorem 5).

Theorem 12 *Let φ be a flat $\mathbf{EATL}^+$ formula, M be a $\mathbf{CGS}$, and q a state in M. There is a polynomial-time alternating Turing machine that returns "yes" if $M, q \models \varphi$ and "no" otherwise.*

Proof. The proof is done analogously to the one of Theorem 5. Now, the verifiers strategy and the first outcome of the opponents is constructed for the first $k := |St_M| \cdot (1 + |\mathcal{APF}(\gamma)| + |\mathcal{APF}^\infty(\gamma)|)$ steps. Then, the game theoretic game to determine a flat atomic subformula is implemented. Finally, this subformula is tested agains the guessed path regarding the semantics $\models^k$. Note, that also also the clause for $\overset{\infty}{\Diamond}$-atomic formulae has to be considered. The correctness follows from Theorem 11. ■

Finally, we get the following result as a combination of Theorem 12 and Theorem 2. The reasoning is exactly the same as for Theorem 6.

Theorem 13 *Model checking $\mathbf{EATL}^+$ with the perfect-recall semantics over $\mathbf{CGS}$'s is $\mathbf{PSPACE}$-complete wrt the size of the model and the length of the formula (even for turn-based models with two agents and flat $\mathbf{ATL}^+$ formulae).*

5. Significance of the Results

Why are the results presented here significant? First of all, we have corrected a widely believed "result" about model checking $\mathbf{ATL}^+$, and that is important on its own. Several other existing claims concerning variants of the model checking problem were based on the $\mathbf{\Delta_3^P}$-completeness for $\mathbf{ATL}^+$, and thus needed to be rectified as well. Moreover, the $\mathbf{ATL}^+$ verification complexity is important because $\mathbf{ATL}^+$ can be seen as the minimal language discerning strategic abilities *with* and *without* memory of past actions. Our results show that the more compact models of agents (which we usually get when perfect memory is assumed) come with a computational price already in the case of $\mathbf{ATL}^+$, and not only for $\mathbf{ATL}^*$ as it was believed before.

$\mathbf{ATL}^+$ deserves attention from the conceptual point of view, too. We argued in Section 2.3 that it enables neat and succinct specifications of sophisticated properties regarding e.g. the outcome of agents' play under behavioral constraints. This is especially clear for $\mathbf{EATL}^+$ where the constraints can take the form of fairness conditions. Constraints of this kind are extremely important when specifying and/or verifying agents in an asynchronous environment, cf. [8]. Since $\mathbf{ATL}^+_{\mathbf{IR}}$ was believed to have the same model checking complexity as $\mathbf{ATL}^+_{\mathbf{Ir}}$, the former seemed a sensible tradeoff between expressivity and complexity. In this context, our new complexity results are rather pessimistic and shift the balance markedly in favor of verification of *memoryless agents*. In consequence, for agents *with* memory one has to fall back to the less expressive logic $\mathbf{ATL_{IR}}$, or accept the less desirable computational properties of $\mathbf{ATL}^+_{\mathbf{IR}}$. On the positive side, we have also shown that fairness properties incur no extra cost in either case and that model checking $\mathbf{ATL}^+/\mathbf{EATL}^+$ is still much cheaper than for $\mathbf{ATL}^*$.

6. Conclusions

In this paper we have corrected a result concerning the model checking complexity of $\mathbf{ATL^+}$ with respect to agents that remember the whole history of the game. In an otherwise excellent study [17], the problem was "proved" to be $\mathbf{\Delta_3^P}$-complete. Our amendment is rather pessimistic as we show that the problem is in fact $\mathbf{PSPACE}$-complete. In consequence, the results on model checking $\mathbf{ATL^+}$, reported in [12], are also incorrect. On the other hand, we also show that adding fairness conditions does not increase the complexity further, which is definitely good news. In consequence, $\mathbf{EATL^+_{IR}}$ is still an interesting option for specification and verification of multi-agent systems if one wants to avoid the prohibitive complexity of model checking with full $\mathbf{ATL^*_{IR}}$.

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[4] N. Bulling and W. Jamroga. Agents, beliefs and plausible behaviour in a temporal setting. In *Proceedings of AAMAS'07*, pages 570–577, 2007.

[5] N. Bulling and W. Jamroga. Model checking ATL$^+$ is harder than it seemed. In *Proceedings of EUMAS'09*, 2009.

[6] N. Bulling and W. Jamroga. Model checking agents with memory is harder than it seemed. In *Proceedings of AAMAS2010*, pages 633–640, 2010.

[7] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.

[8] M. Dastani and W. Jamroga. Reasoning about strategies of multi-agent programs. In *Proceedings of AAMAS2010*, pages 625–632, 2010.

[9] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[10] A. Harding, M. Ryan, and P.-Y. Schobbens. Approximating ATL* in ATL. In *VMCAI '02: Revised Papers from the Third International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 289–301. Springer-Verlag, 2002.

[11] J. Hintikka. *Logic, Language Games and Information*. Clarendon Press : Oxford, 1973.

[12] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *LMCS*, 4:7, 2008.

[13] F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking CTL+ and FCTL is hard. In *Proceedings of FoSSaCS'01*, volume 2030 of *Lecture Notes in Computer Science*, pages 318–331. Springer, 2001.

[14] A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.

[15] A. Lomuscio and M. Sergot. A formalisation of violation, error recovery, and enforcement in the bit transmission problem. *Journal of Applied Logic*, 2(1):93–116, 2004.

[16] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley : Reading, 1994.

[17] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.

[18] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility and synthesis. *Synthese*, 156(1):1–19, 2005.

[19] T. Wilke. CTL+ is exponentially more succint than CTL. In *Proceedings of FST&TCS '99*, volume 1738 of *LNCS*, pages 110–121, 1999.

[20] B. Wozna and A. Lomuscio. A logic for knowledge, correctness, and real time. In *Proceedings of CLIMA V*, Lecture Notes in Computer Science. Springer, 2004.