Verifying Agents with Memory Is Harder than It Seemed

Nils Bulling
Institute of Computer Science
Clausthal University of Technology
bulling@in.tu-clausthal.de

Wojciech Jamroga
Computer Science and Communication
University of Luxembourg
wojtek.jamroga@uni.lu

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 $\mathbf{ATL}^*$.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Modal logic*

General Terms

Theory, Verification

Keywords

Strategic logic, model checking, complexity

1. INTRODUCTION

The alternating-time temporal logic $\mathbf{ATL}^*$ and its less expressive version **ATL** [2] 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. 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 [2], i.e., if players can memorize the whole history of the game. Hence, the latter 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 will be accomplished.

Verification with $\mathbf{ATL}^+$ has been believed to be $\mathbf{\Delta_3^P}$-complete for both memoryless and perfect recall strategies [15]. 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 $\mathbf{EATL}^+$ ($\mathbf{ATL}^+$ augmented with the temporal operator $\Box\Diamond$ ("infinitely often")). Finally, we argue why we consider the results significant and present some conclusions in Sections 5 and 6, respectively.

2. ATL$^+$ AND THE MATTER OF RECALL

The *alternating-time temporal logic* $\mathbf{ATL}^*$ [2] is a temporal logic that incorporates some basic game-theoretical notions. Essentially, $\mathbf{ATL}^*$ generalizes the branching time logic $\mathbf{CTL}^*$ [5] 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

Cite as: Verifying Agents with Memory Is Harder than It Seemed, Nils Bulling and Wojciech Jamroga, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 699-706

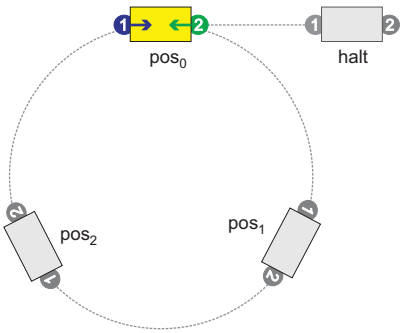

Figure 1: Two robots and a carriage: a schematic view

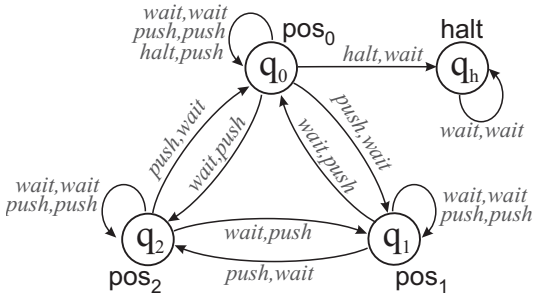

Figure 2: Two robots and a carriage: concurrent game structure M_1 that models the scenario

set of *agents*. Alternating-time temporal logic comes in several variants, of which **ATL*** is the broadest. Formally, the language of **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 **ATL***.

The best known variant of the alternating time temporal logics is **ATL** (sometimes called "**ATL** without star" or "vanilla" **ATL**) in which every occurrence of a cooperation modality is immediately followed by exactly one temporal operator. In this paper, however, we study the model checking problem for **ATL**$^+$, a variant that sits between **ATL*** and **ATL**. The language of **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 $\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc\varphi \mid \varphi \mathcal{U} \varphi$.

Example 1. *The* **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* **ATL*** *formula which clearly belongs to neither* **ATL** *nor* **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* **ATL**$^+$ *but not of* **ATL**.

2.2 Semantics

The semantics of **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* (**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 $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$, 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 $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to state q and a tuple of actions $\langle \alpha_1, \ldots, \alpha_k \rangle$ 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.

A *path* $\lambda = q_0 q_1 q_2 \ldots$ is an infinite sequence of states such that there is a transition between each q_i, q_{i+1}. We use $\lambda[i]$ 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.*

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. To distinguish between those cases, we use the taxonomy and notation introduced in [15]: $\mathbf{ATL_{xy}}$ 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 *IR*-setting.

A *perfect information perfect recall strategy* (*IR-strategy*) for agent a is a function $s_a : St^+ \rightarrow Act$ such that $s_a(q_0 q_1 \ldots q_n) \in d_a(q_n)$ for any finite history $q_0 q_1 \ldots q_n$. A *perfect information memoryless strategy* (*Ir-strategy*) is a function $s_a : St \rightarrow Act$ 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 $A = \{a_1, \ldots, a_r\} \subseteq \mathbb{A}\mathrm{gt}$ is simply a tuple of individual strategies $s_A = \langle s_{a_1}, \ldots, s_{a_r} \rangle$.

By $s_A|_a$, we denote agent a's part s_a of the collective strategy s_A, $a \in A$. 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:

$out(q, s_A) = \{\lambda = q_0 q_1 q_2 \ldots \mid q_0 = q$ and for each $i = 1, 2, \ldots$ there exists a tuple of agents' decisions $\langle \alpha_{a_1}^{i-1}, \ldots, \alpha_{a_k}^{i-1} \rangle$ such that $\alpha_a^{i-1} \in d_a(q_{i-1})$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha_a^{i-1} = s_A|_a(q_0 q_1 \ldots q_{i-1})$ for every $a \in A$, and $o(q_{i-1}, \alpha_{a_1}^{i-1}, \ldots, \alpha_{a_k}^{i-1}) = q_i\}$.

The definition for memoryless strategies is analogous.

Let M be a **CGS**, q a state, and λ a path in M. The semantics of $\mathbf{ATL^*_{xy}}$ is defined as follows [2, 15]:

- $M, q \models \mathsf{p}$ 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 A such that for each $\lambda \in out(s_A, q)$ 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$^+$. Known Results

It is well known that the memoryless and perfect recall semantics of "vanilla" **ATL** coincide [2, 15]. 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 a minimal well-known variant of the alternating-time logic that discerns the memoryless and perfect recall semantics.

In conceptual terms, we can use $\mathbf{ATL}^+$ to specify a set of goals 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. This kind of properties has been especially studied in deontic logic and normative systems (e.g., [12, 13, 17]), but also in reasoning about plausible behavior of agents [3]. For instance, consider a class of systems where a state is labeled with proposition V_a iff agent a has violated a social norm with his last action. Then, property "a can enforce property γ while complying with social norms" can be captured 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 $\langle\langle b \rangle\rangle ((\Box \neg \mathsf{V}_a) \rightarrow \gamma)$.

$\mathbf{ATL}^+$ is more expressive than **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 **ATL** [10]. Still, many formulae of $\mathbf{ATL}^+$ have their equivalent counterparts in **ATL**. For instance, $\langle\langle jamesbond \rangle\rangle (\Box \neg \mathsf{crash} \wedge \Diamond \mathsf{land})$ can be rephrased 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 $\mathbf{ATL^+_{IR}}$ formulae can be equivalently translated into $\mathbf{ATL_{IR}}$ *with the "weak until" operator* [8]. We observe that in some cases the translation results in an exponential blowup of the length of the formula. Thus, $\mathbf{ATL^+_{IR}}$ has the same expressive power as "vanilla" $\mathbf{ATL_{IR}}$ with "weak until" – but it allows for exponentially more succinct and intuitive specifications of some properties (this follows from the results in [16]).

Regarding the complexity of verification for strategic logics, the following patterns can be observed (albeit not without exceptions): (i) Model checking more expressive and/or succinct logics is usually harder than the less expressive and/or succinct ones; (ii) Model checking imperfect information agents is usually harder than perfect information ones; (iii)Model checking agents with perfect recall is usually harder than memoryless agents.

Indeed, for the memoryless semantics (Ir), model checking of **ATL** can be done in linear time wrt the number of transitions in the model and the length of the formula [2],[1] while model checking of $\mathbf{ATL}^+$ is $\mathbf{\Delta_3^P}$-complete,[2] and model checking of $\mathbf{ATL}^*$ is **PSPACE**-complete. Moreover, for the perfect recall semantics (IR), model checking of **ATL** is still linear (it is *the same* logic after all) while verification of $\mathbf{ATL}^*$ is complete in double exponential time [2].

What about model checking $\mathbf{ATL^+_{IR}}$? In [15], it is claimed to be $\mathbf{\Delta_3^P}$-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 **PSPACE**-complete in the IR-setting. Note that the results in [10] on the verification of $\mathbf{ATL^+_{IR}}$ in various non-standard settings are also incorrect since they crucially depend on the claim from [15]; we will correct them in Section 3.3.

3. MODEL CHECKING ATL$^+_{IR}$

In an excellent study [15], Schobbens claims that model checking $\mathbf{ATL}^+$ is $\mathbf{\Delta_3^P}$-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, and model-check the $\mathbf{CTL}^+$

[1] It is important to add that the "weak until" operator $\mathcal{W}$ does not increase the complexity [10].

[2] $\mathbf{\Delta_3^P} = \mathbf{P^{\Sigma_2^P}}$ is the class of problems that can be solved by a deterministic Turing machine that can make adaptive queries to an oracle of type $\mathbf{\Sigma_2^P} = \mathbf{NP^{NP}}$. 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.

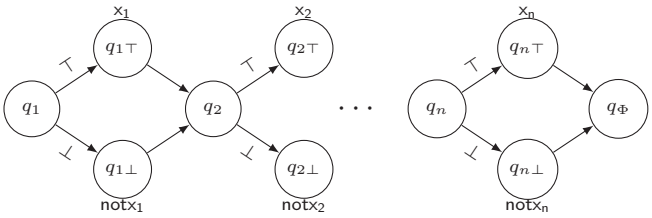

Figure 3: Construction of the concurrent game structure for QSAT: value choice section

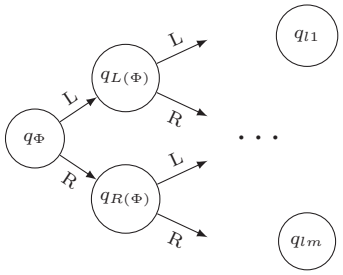

Figure 4: CGS for QSAT: formula structure section

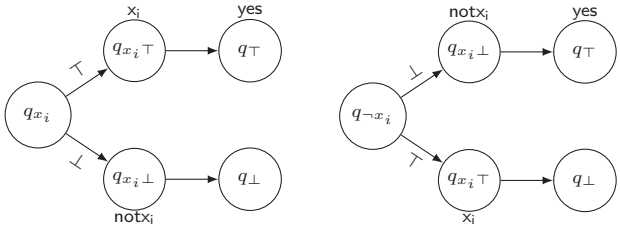

Figure 5: CGS for QSAT: sections of literals

formula $\mathsf{A}\gamma$ in the resulting model. 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 $\mathsf{A}\gamma$ can be done in nondeterministic polynomial time [11], we get that the overall procedure runs in time $\left(\mathbf{\Delta_2^P}\right)^{\mathbf{NP}} = \mathbf{\Delta_3^P}$ [15].

For agents with perfect recall, a similar argument *seems* correct. Every formula of $\mathbf{ATL^+_{IR}}$ can be translated to an equivalent formula of $\mathbf{ATL_{IR}}$ with weak until [8], and for **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 $\mathbf{ATL^+_{IR}}$ formula). For example, formula $\langle\langle A\rangle\rangle(\Diamond\mathsf{p} \wedge \Diamond\mathsf{q})$ is translated to $\langle\langle A\rangle\rangle\Diamond\big((\mathsf{p} \wedge \langle\langle A\rangle\rangle\Diamond\mathsf{q}) \vee (\mathsf{q} \wedge \langle\langle A\rangle\rangle\Diamond\mathsf{p})\big)$; for a longer list of achievement goals ($\Diamond\mathsf{p}_i$) 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 whom 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 $\mathbf{ATL^+_{IR}}$ in the end? The problem turns out to be harder than $\mathbf{\Delta_3^P}$, namely **PSPACE**-complete.

3.1 Lower Bound

We prove the **PSPACE**-hardness by a reduction of Quantified Boolean Satisfiability (QSAT), a canonical **PSPACE**-complete problem.

Definition 1 (QSAT [14]). ***Input:*** *A Boolean formula Φ in negation normal form (i.e., negations occur only at literals) with n propositional variables $x_1, \ldots, x_n$.*
Output: *True if $\exists x_1 \forall x_2 \ldots Q_n x_n\ \Phi$ holds, 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[3] concurrent game structure M with two players: the *verifier* **v** and the *refuter* **r**. The structure consists of the following sections:

- *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 **v**, states with even i are controlled by **r**. The owner of a state can choose between two possible valuations $(\top, \bot)$. Choosing $\top$ leads to a state where the proposition $\mathsf{x_i}$ holds; choosing $\bot$ leads to a state labeled by the 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 $\bot$) 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 $\bot$). 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 $q_\bot$ otherwise – see Figure 5 for details.

[3] A model is *turn-based* if each state has 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).

Note that the value of variable x_i can be declared twice during an execution of the model. 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 $\bot$ during a single execution. Now, we have that:

LEMMA 1. *$\exists x_1 \forall x_2 \ldots Q_n x_n\ \Phi$ iff*

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

PROOF SKETCH. 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 $x_1, \ldots, x_{i-1}$. Note that this is the reason why perfect recall is necessary to obtain the reduction. □

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 **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* **PSPACE***-hard with respect to the size of the model and the length of the formula. It is* **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 [11], 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 SKETCH. 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. 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 sequences of states.

DEFINITION 4 ($\models^k$). *Let M be a* **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 $0 \leq j < i$;

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

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 SKETCH. "⇒:" Let s_A be a strategy for (M, q, γ). By Lemma 3 and the fact that $|wit(\lambda, \gamma)| \leq |\mathcal{APF}(\gamma)|$ for any path λ there is a strategy s'_A for (M, q, γ) such that $\max wit(\lambda, \gamma) \leq |St_M| \cdot |\mathcal{APF}(\gamma)|$ 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).

"⇐": 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 $j \leq v$, 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) $w := witpos(\lambda, \gamma') \geq k$ that cannot be prevented by A. Due to efficiency all subformulae that have a witness $\leq k$ actually have a witness $\leq k - |St_M|$. 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 $\leq v$ (i.e. within steps v and k). This contradicts that s_A is a k-witness strategy. □

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* **CGS***, and q a state. Then, there is a polynomial-time alternating Turing machine (wrt the size of the model and length of the formula) that returns "yes" if $M, q \models \varphi$ and "no" otherwise.*

PROOF SKETCH. 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.[4] Next, we allow the verifier to nondeterministically construct A's strategy step by step for the first $|St_M| \cdot |\mathcal{APF}(\gamma)|$ rounds ($|\mathbb{A}\text{gt}|$ steps each), while the refuter guesses the most damaging responses of $\mathbb{A}\text{gt} \setminus A$. 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 [9] 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 non-flat formulae we proceed as usual (cf. [2]).

COROLLARY 6. *Model checking* $\mathbf{ATL}^+$ *over* **CGS***'s with the perfect recall semantics is* **PSPACE***-complete wrt the size of the model and the length of the formula. It is* **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 $\mathbb{A}\text{gt}$. 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* (**ATS**), used as models in the initial semantics of **ATL** [1], agents' choices are state transformations themselves rather than abstract labels. In *implicit concurrent game structures* [10], the transition array is defined by Boolean expressions. **ATS** and implicit **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" **CGS**. In [10], Laroussinie et al. claim that model checking $\mathbf{ATL}^+_{\mathbf{IR}}$ against **ATS** as well as implicit **CGS** is $\mathbf{\Delta_3^P}$-complete. Since the proofs are actually based on the flawed result from [15], 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 **ATS** is a tuple $M = \langle \mathbb{A}\text{gt}, St, \Pi, \pi, \delta \rangle$, where $\mathbb{A}\text{gt}, St, \Pi, \pi$ are like in a **CGS**, and $\delta : St \times \mathbb{A}\text{gt} \to 2^{2^{St}}$ 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 $a \in \mathbb{A}\text{gt}$. 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 CGS. An implicit **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 in [10, Section 3.4.1], where model checking $\mathbf{ATL}^+$ with respect to both **ATS** and implicit **CGS** is claimed to be $\mathbf{\Delta_3^P}$-complete, we establish the complexity as **PSPACE**.

THEOREM 7. *Model checking* $\mathbf{ATL}^+$ *for* **ATS** *and implicit* **CGS** *using the perfect recall semantics is* **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 **CGS** is linear in the number of states (n), agents (k), and actions (d). Moreover, each turn-based **CGS** has an isomorphic **ATS**, and an isomorphic implicit **CGS**; the transformation takes $O(nd)$ steps.

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

This, together with the reduction from Section 3.1, gives us **PSPACE**-hardness wrt n, k, d and the length of the formula (l) for model checking $\mathbf{ATL}^+$ against **ATS** as well as implicit **CGS**.

Upper bound. A close inspection of the algorithm from Section 3.2 reveals that it can be as well applied to **ATS** and implicit **CGS**. □

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 $A \subsetneq \mathbb{A}\mathrm{gt}$ 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.

4.1 EATL$^+$

$\mathbf{EATL}^+$ is a subset of $\mathbf{ATL}^*$ obtained by extending $\mathbf{ATL}^+$ path formulae: $\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \Box\Diamond\varphi$.

We note that $\mathbf{EATL}^+$ is strictly more expressive than **ATL** with "weak until" (it follows from the fact that **ECTL** is strictly more expressive than **CTL** [7]), and hence also more expressive than $\mathbf{ATL}^+$. Moreover, in order to reason about the outcome of fair computations in model M, it suffices to do the following. First, we add to M special propositions act_i, one per agent i. The propositions indicate which agent has executed the most recent action. Now, for example, the $\mathbf{EATL}^+$ $\langle\!\langle 1,2\rangle\!\rangle\big((\bigwedge_i \Box\Diamond\mathsf{act}_i) \rightarrow \Diamond\mathsf{cleanRoom}\big)$ 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 $\gamma \equiv \Box\Diamond\varphi$. If $\lambda \not\models \Box\Diamond\varphi$ then $wit^\infty(\lambda,\gamma) = \emptyset$; and if $\lambda \models \Box\Diamond\varphi$ then $wit^\infty(\lambda,\gamma) = \{i \mid \lambda[i] \models \varphi\}$. Note that the set is either infinite or empty.

Moreover, an $\mathbf{ATL}^+$-path formula γ is called $\Box\Diamond$-*atomic* if it has the form $\Box\Diamond\varphi_1$. For $\varphi \in \mathbf{ATL}^+$ we denote the set of all $\Box\Diamond$-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 $\Box\Diamond$-atomic flat formula is true. The following proofs are done similarly to the ones given in Section 3.2 and can be found in [4].

Proposition 8. *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 $i \geq t$.*

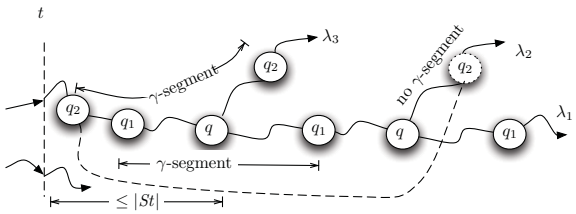

Figure 6: Proof idea of Theorem 10.

Lemma 9. *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.*

In the following we extend the finite path semantics such that it can deal with $\Box\Diamond$-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: $M,\lambda \models^k \Box\Diamond\gamma$ 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 $M,\lambda \models^k \gamma$.

Theorem 10. *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. "⇒": 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 8 and Lemma 9. 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 6) resulting in a $|St_M| \cdot (1 + |\mathcal{APF}(\gamma)| + |\mathcal{APF}^\infty(\gamma)|)$-witness strategy for (M,q,γ).

"⇐": 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 6). 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. 6) 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 $M, \lambda_2 \not\models \gamma'$. $\square$

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 11. *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 returns "yes" if $M, q \models \varphi$ and "no" otherwise.*

Finally, we get the following result as a combination of Theorem 11 and Theorem 2.

THEOREM 12. *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 agents in an asynchronous environment, cf. [6]. 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 [15], 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 [10], 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, $\mathbf{EATL}^+_{\mathbf{IR}}$ is still an interesting option for specification and verification of MAS if one wants to avoid the prohibitive complexity of model checking with full $\mathbf{ATL}^*_{\mathbf{IR}}$.

7. REFERENCES

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

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

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

[4] N. Bulling and W. Jamroga. Model checking ATL$^+$ is harder than it seemed. Technical Report IfI-09-13, Clausthal University of Technology, 2009.

[5] E. Clarke and E. 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.

[6] M. Dastani and W. Jamroga. Reasoning about strategies of multi-agent programs. In *Proceedings of AAMAS2010*, 2010.

[7] 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.

[8] A. Harding, M. Ryan, and P. 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.

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

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

[11] F. Laroussinie, N. Markey, and P. 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.

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

[13] 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.

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

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

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

[17] 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.