ATL* Satisfiability is 2EXPTIME-Complete[⋆]

Sven Schewe

Universität des Saarlandes, 66123 Saarbrücken, Germany

Abstract. The two central decision problems that arise during the design of safety critical systems are the satisfiability and the model checking problem. While model checking can only be applied after implementing the system, satisfiability checking answers the question whether a system that satisfies the specification exists. Model checking is traditionally considered to be the simpler problem – for branching-time and fixed point logics such as CTL, CTL*, ATL, and the classical and alternating time μ-calculus, the complexity of satisfiability checking is considerably higher than the model checking complexity. We show that ATL* is a notable exception of this rule: Both ATL* model checking and ATL* satisfiability checking are 2EXPTIME-complete.

1 Introduction

One of the main challenges in system design is the construction of correct implementations from their temporal specifications. Traditionally, system design consists of three separated phases, the specification phase, the implementation phase, and the validation phase. From a scientific point of view it seems inviting to overcome the separation between the implementation and validation phase, and replace the manual implementation of a system and its subsequent validation by a push-button approach, which automatically synthesizes an implementation that is correct by construction. Automating the system construction also provides valuable additional information: we can distinguish *unsatisfiable* system specifications, which otherwise would go unnoticed, leading to a waste of effort in the fruitless attempt of finding a correct implementation.

One important choice on the way towards the ideal of fully automated system construction is the choice of the specification language. For temporal specifications, three different types of logics have been considered: Linear-time logics [1], branching-time logics [2], and alternating-time logics [3]. The different paradigms are suitable for different types of systems and different design phases. Linear-time logic can only reason about properties of all possible runs of the system. Consequently, it cannot express the existence of different runs. A constructive non-emptiness test of an LTL specification is therefore bound to create a system that has exactly one possible run. Branching-time logics [2], on the other hand,

⋆ This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

Logic	Model Checking (Structure)	Model Checking	Satisfiability Checking
LTL	NLOGSPACE [4]	PSPACE [5]	PSPACE [4]
CTL	NLOGSPACE [6]	PTIME [5]	EXPTIME [2]
CTL*	NLOGSPACE [6]	PSPACE [5]	2EXPTIME [4]
ATL	PTIME [3]	PTIME [3]	EXPTIME [7]
ATL*	PTIME [3]	2EXPTIME [3]	**2EXPTIME**

Fig. 1. For all previously considered branching-time temporal specifications, satisfiability checking is at least exponentially harder than model checking (in the specification). We show that ATL* is an interesting exception to this rule.

can reason about *possible* futures, but they refer to *closed systems* [3] that do not distinguish between different participating agents. Finally, alternating-time logics [3] can reason about the strategic capabilities of groups of agents to cooperate to obtain a temporal goal. The following example illustrates the differences.

Consider a vending machine that offers coffee and tea. Ultimately, we want the machine to react on the requests of a customer, who shall be provided with coffee or tea upon her wish. In alternating-time logic, we have a natural correspondence to this requirement: we simply specify that the customer has the strategic capability to get coffee or tea from the machine, without the need of cooperation, written $\langle\langle\text{customer}\rangle\rangle \bigcirc \text{get}_{\mathit{coffee}}$ or $\langle\langle\text{customer}\rangle\rangle \bigcirc \text{get}_{\mathit{tea}}$, respectively.

In branching-time logic, there is no natural correspondence to the property. The typical approximation is to specify the *possibility* that coffee or tea is provided, written $E \bigcirc \text{get}_{\mathit{coffee}}$ or $E \bigcirc \text{get}_{\mathit{tea}}$, respectively. However, this does no longer guarantee that the customer can choose; the specification is also fulfilled if the health insurance company can override the decision for coffee. A workaround may be to introduce a dedicated communication interface between the vending machine and the customer, and represent the desire for coffee by a $\text{desire}_{\mathit{coffee}}$ bit controlled by the customer. The property may then be approximated by $E \bigcirc \text{desire}_{\mathit{coffee}}$ and $\text{desire}_{\mathit{coffee}} \rightarrow A \bigcirc \text{get}_{\mathit{coffee}}$. In LTL, the possibility of different system behaviors cannot be expressed within the logic. Here, in addition to specifying an interface, we would have to distinguish between parts of the system under our control (the vending machine) and parts outside of our control (the customer). The most likely approximation would be $\text{desire}_{\mathit{coffee}} \rightarrow \bigcirc \text{get}_{\mathit{coffee}}$, with the addition that there is not only the need to design an interface to the customer beforehand, but also to make assumptions about her behavior.

Using the workarounds for branching-time or linear-time logic requires solving the central design problem of designing interfaces in an early specification phase, instead of starting with an abstract view on the system. Especially in the case that synthesis fails, we could no longer distinguish if we made an error in designing the interfaces, or if the specification is unrealizable.

As an example for this effect, we could consider to use alternating-time logic for the specifications of protocol fairness. ATL* has, for example, been used to express the fairness requirement "Bob cannot obtain Alice's signature unless Alice can obtain Bob's signature as well" [8] in contract signing protocols. Using satisfiability checking techniques for alternating-time logics, we can automate [9] the proof that fair contract signing is not possible without a trusted third party [10] (under the presence of other standard requirements).

The alternating-time temporal logic ATL* [3] extends the classic branching-time temporal logic CTL* [4] with path quantifiers that refer to the strategic capabilities of groups of agents. An ATL* specification $\langle\langle A' \rangle\rangle \varphi$ requires that the group A' of agents can cooperate to enforce the path formula φ. ATL* formulas are interpreted over concurrent game structures, a special type of labeled transition systems, where each transition results from a set of decisions, one for each agent. When interpreted over a concurrent game structure $\mathcal{C}$, $\langle\langle A' \rangle\rangle \varphi$ holds true in a state s of $\mathcal{C}$ if the agents in A' can win a two player game against the agents not in A'. In this game, the two groups of agents take turns in making their decisions (starting with the agents in A'), resulting in an infinite sequence $ss_1s_2\ldots$ of states of the concurrent game structure $\mathcal{C}$. The agents in A' win this game, if the infinite sequence $ss_1s_2\ldots$ satisfies the path formula φ.

Since ATL* specifications can canonically be transformed into alternating-time μ-calculus (ATM) formulas [11, 3], ATL* inherits the decidability and finite model property from ATM [9]. This translation from ATL* to ATM, comprises a doubly exponential blow-up, which is in line with the doubly exponential model checking complexity of ATL* [11, 3]. The *complexity* of the ATL* satisfiability and synthesis problem, on the other hand, has been an interesting open challenge since its introduction [7]: While the complexity of the satisfiability problem is known to be EXPTIME-complete for the least expressive alternating-time logic ATL [12, 7] as well as for the most expressive alternating-time logic ATM [9], the complexity of the succinct and intuitive temporal logic ATL* has only been known to be in 3EXPTIME [9, 11], and to inherit the 2EXPTIME hardness from CTL*, leaving an exponential gap between both bounds.

Outline. In this paper, we introduce an automata-theoretic decision procedure to demonstrate that deciding the satisfiability of an ATL* specification and, for satisfiable specifications, constructing a correct model of the specifications is no more expensive than model checking: both problems are 2EXPTIME-complete in the size of the specification. To the contrary, the cost of model checking a concurrent game structure against an ATL* specification is also polynomial in the size of the concurrent game structure. While polynomial conveys the impression of feasibility, the degree of this polynomial is, for known algorithms, exponential in the size of the specification [3, 11].

On first glance, an automata-theoretic construction based on automata over concurrent game structures (ACGs) [9] – the alternating-time extension of symmetric alternating-automata [13] – does not seem to be a promising starting point for the construction of a 2EXPTIME algorithm, because synthesis procedures based on alternating automata usually shift all combinatorial difficulties to testing their non-emptiness [14]. Using a doubly exponential translation from ATL* through ATM to an equivalent ACG suffices to proof the finite model property of ATL* [9], but indeed leads to a *triply* exponential construction.

In order to show that a constructive non-emptiness test for ATL* specifications can be performed in doubly exponential time, we combine two concepts: We first show that every model can be transformed into an *explicit* model that includes a certificate of its correctness. For this special kind of model, it suffices to build an ACG that only checks the correctness of the certificate. Finally, we show that we can construct such an automaton, which is only singly exponential in the size of the specification. Together with the exponential cost of a constructive non-emptiness test of ACGs [9], we can provide a 2EXPTIME synthesis algorithm for ATL* specifications that returns a model together with a correctness certificate. 2EXPTIME-completeness then follows with the respective hardness result for the syntactic sublogic CTL* [4] of ATL*.

2 Logic, Models and Automata

In this section we recapture the logic ATL* [3], concurrent game structures, over which ATL* specifications are interpreted, and automata over concurrent game structures [9], which are used to represent alternating-time specifications.

2.1 Concurrent Game Structures

Concurrent game structures [3] generalize labeled transition systems (or pointed Kripke structures) to a setting with multiple agents. A *concurrent game structure* (CGS) is a tuple $\mathcal{C} = (P, A, S, s_0, l, \Delta, \tau)$, where

- P is a finite nonempty set of atomic propositions,
- A is a finite nonempty set of agents,
- S is a nonempty set of states, with a designated initial state $s_0 \in S$,
- $l : S \to 2^P$ is a labeling function that decorates each state with a subset of the atomic propositions,
- Δ is a nonempty set of possible decisions for every agent, and
- $\tau : S \times \Delta^A \to S$ is a transition function that maps a state and the decisions of the agents to a new state.

For a CGS $\mathcal{C}$, a *strategy* for a set $A' \subseteq A$ of agents is a mapping $f_{A'} : S^* \to \Delta^{A'}$ from finite traces to decisions of the agents in A', and a *counter strategy* is a mapping $f^c_{A \smallsetminus A'} : S^* \times \Delta^{A'} \to \Delta^{A \smallsetminus A'}$ from finite traces and decisions of the agents in A' to decisions of the agents in $A \smallsetminus A'$. For a given strategy $f_{A'}$ and counter strategy $f^c_{A \smallsetminus A'}$, the set of *plays* starting at a position s_1 is defined as
$$\mathit{plays}(s_1, f_{A'}) = \{s_1 s_2 s_3 \ldots \mid \forall i{\geq}1\, \exists d'{\in}\Delta^{A \smallsetminus A'}.\, s_{i+1} = \tau(s_i, (f_{A'}(s_1 \ldots s_i), d'))\},$$
$$\mathit{plays}(s_1, f^c_{A \smallsetminus A'}) = \{s_1 s_2 s_3 \ldots \mid \forall i{\geq}1\, \exists d{\in}\Delta^{A'}.\, s_{i+1}{=}\tau(s_i, (f^c_{A \smallsetminus A'}(s_1 \ldots s_i, d), d))\}.$$

2.2 ATL*

ATL* extends the classical branching-time logic CTL* by path quantifiers that allow for reasoning about the strategic capability of groups of agents.

ATL* Syntax. ATL* contains formulas $\langle\!\langle A' \rangle\!\rangle \psi$, expressing that the group $A' \subseteq A$ of agents can enforce that the path formula ψ holds true. Formally, the state formulas (Φ) and path formulas (Π) of ATL* are given by the following grammar (where $p \in P$ is an atomic proposition, and $A' \subseteq A$[1] is a set of agents).

$$\begin{aligned}\Phi &:= \mathit{true} \mid p \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi \mid \langle\!\langle A' \rangle\!\rangle \Pi, \text{ and}\\ \Pi &:= \Phi \mid \Pi \wedge \Pi \mid \Pi \vee \Pi \mid \neg\Pi \mid \bigcirc\Pi \mid \Pi \,\mathrm{U}\, \Pi.\end{aligned}$$

Every state formula is an ATL* formula. We call an ATL* formula *basic* iff it starts with a path quantifier $\langle\!\langle A' \rangle\!\rangle$.

Semantics. An ATL* specification with atomic propositions $\subseteq P$ is interpreted over a CGS $\mathcal{C} = (P, A, S, s_0, l, \Delta, \tau)$. $\|\varphi\|_\mathcal{C} \subseteq S$ denotes the set of states where φ holds. A CGS $\mathcal{C} = (P, A, S, s_0, l, \Delta, \tau)$ is a *model* of a specification φ ($\mathcal{C} \models \varphi$) with atomic propositions P iff φ holds in the initial state ($s_0 \in \|\varphi\|_\mathcal{C}$).

For each state s of $\mathcal{C}$, $\mathit{path}(s)$ denotes all paths in $\mathcal{C}$ that originate from s, and $\mathit{path}(\mathcal{C}) = \bigcup\{\mathit{path}(s) \mid s \in S\}$ denotes the set of all paths in $\mathcal{C}$.

An ATL* formula is evaluated along the structure of the formula.

- Atomic propositions and Boolean connectives are interpreted as usual: $\|\mathit{true}\|_\mathcal{C} = S$, $\|p\|_\mathcal{C} = \{s \in S \mid p \in l(s)\}$, and $\|\varphi \wedge \psi\|_\mathcal{C} = \|\varphi\|_\mathcal{C} \cap \|\psi\|_\mathcal{C}$, $\|\varphi \vee \psi\|_\mathcal{C} = \|\varphi\|_\mathcal{C} \cup \|\psi\|_\mathcal{C}$, and $\|\neg\varphi\|_\mathcal{C} = S \smallsetminus \|\varphi\|_\mathcal{C}$.
- Basic formulas $\varphi = \langle\!\langle A' \rangle\!\rangle\psi$ hold true in a state s if the agents in A' have a strategy which ensures that all plays starting in s satisfy the path formula ψ: $s \in \|\varphi\|_\mathcal{C} \Leftrightarrow \exists f_{A'} : S^* \to \Delta^{A'}.\, \mathit{plays}(s, f_{A'}) \subseteq \|\psi\|_\mathcal{C}^{\mathit{path}}$.

For a path formula φ and a CGS $\mathcal{C}$, $\|\varphi\|_\mathcal{C}^{\mathit{path}} \subseteq \mathit{path}(\mathcal{C})$ denotes the set of paths of $\mathcal{C}$ where φ holds. Path formulas are interpreted as follows:

- For state formulas φ, $\|\varphi\|_\mathcal{C}^{\mathit{path}} = \bigcup\{\mathit{path}(s) \mid s \in \|\varphi\|_\mathcal{C}\}$.
- Boolean connectives are interpreted as usual: $\|\varphi\wedge\psi\|_\mathcal{C}^{\mathit{path}} = \|\varphi\|_\mathcal{C}^{\mathit{path}} \cap \|\psi\|_\mathcal{C}^{\mathit{path}}$, $\|\varphi \vee \psi\|_\mathcal{C}^{\mathit{path}} = \|\psi\|_\mathcal{C}^{\mathit{path}} \cup \|\varphi\|_\mathcal{C}^{\mathit{path}}$, and $\|\neg\varphi\|_\mathcal{C}^{\mathit{path}} = \mathit{path}(\mathcal{C}) \smallsetminus \|\varphi\|_\mathcal{C}^{\mathit{path}}$.
- A path $\pi = s_1, s_2, s_3, s_4 \ldots$ satisfies $\bigcirc\varphi$ (read: next φ), if the path $s_2, s_3, s_4 \ldots$ obtained by deleting the first letter of π satisfies φ: $\|\bigcirc\varphi\|_\mathcal{C}^{\mathit{path}} = \{s_1, s_2, s_3, s_4 \ldots \in \mathit{path}(\mathcal{C}) \mid s_2, s_3, s_4 \ldots \in \|\varphi\|_\mathcal{C}^{\mathit{path}}\}$.
- A path $\pi = s_1, s_2, s_3, s_4 \ldots$ satisfies $\varphi \,\mathrm{U}\, \psi$ (read: φ until ψ), if there is a natural number $n \in \mathbb{N}$ such that
  (1) the path $s_n, s_{n+1}, s_{n+2} \ldots$ obtained by deleting the initial sequence $s_1, s_2, s_3 \ldots s_{n-1}$ of π satisfies the path formula ψ, and
  (2) for all $i < n$, the path $s_i, s_{i+1}, s_{i+2} \ldots$ obtained by deleting the initial sequence $s_1, s_2, s_3 \ldots s_{i-1}$ of π satisfies the path formula φ:
  $\|\varphi \,\mathrm{U}\, \psi\|_\mathcal{C}^{\mathit{path}} = \{s_1, s_2, s_3, s_4 \ldots \in \mathit{path}(\mathcal{C}) \mid$
  $\exists n \in \mathbb{N}.\, (s_n, s_{n+1}, s_{n+2} \ldots \in \|\psi\|_\mathcal{C}^{\mathit{path}} \wedge \forall i < n.\, s_i, s_{i+1}, s_{i+2} \ldots \in \|\varphi\|_\mathcal{C}^{\mathit{path}})\}$.

Note that the validity of basic formulas $\langle\!\langle A' \rangle\!\rangle\psi$ is implicitly defined by the outcome of a two player game with an ω-regular (LTL) objective. Such games are determined [11]. Consequently, there is a counter strategy $f^c_{A \smallsetminus A'} : S^* \times \Delta^{A'} \to \Delta^{A \smallsetminus A'}$ such that $\mathit{plays}(s, f^c_{A \smallsetminus A'}) \subseteq \|\neg\psi\|_\mathcal{C}^{\mathit{path}}$ if and only if $s \notin \|\langle\!\langle A' \rangle\!\rangle\psi\|_\mathcal{C}$.

[1] We assume that the set A of agents is known and fixed. For satisfiability checking, one could argue that this is not necessarily the case. However, we can assume without loss of generality that there is at most one agent that does not occur in the formula [7].

2.3 Automata over Concurrent Game Structures

Automata over concurrent game structures (ACGs) [9] provide an automata-theoretic framework for alternating-time logics. Generalizing symmetric automata [13], ACGs contain *universal atoms* $(\Box, A')$, which refer to *all* successor states for *some* decision of the agents in A', and *existential atoms* $(\Diamond, A')$, which refer to *some* successor state for *each* decision of the agents *not* in A'. ACGs can run on CGSs with arbitrary sets $\varDelta$ of decisions. For the purpose of this paper, it suffices to consider ACGs with a Co-Büchi acceptance condition.

An ACG is a tuple $\mathcal{A} = (\varSigma, Q, q_0, \delta, F)$, where $\varSigma$ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is a designated initial state, δ is a transition function, and $F \subseteq Q$ is a set of final states. The transition function $\delta : Q \times \varSigma \to \mathbb{B}^+(Q \times (\{\Box, \Diamond\} \times 2^A))$ maps a state and an input letter to a positive Boolean combination of universal atoms – $(q, \Box, A')$ – and existential atoms – $(q, \Diamond, A')$.

A run tree $\langle R, r : R \to Q \times S\rangle$ on a given CGS $\mathcal{C} = (P, A, S, s_0, l, \varDelta, \tau)$ is a $Q \times S$-labeled tree where the root is labeled with (q_0, s_0) and where, for a node n with a label (q, s) and a set $L = \{r(n \cdot \rho) \mid n \cdot \rho \in R\}$ of labels of its successors, there is a set $\mathfrak{A} \subseteq Q \times (\{\Box, \Diamond\} \times 2^A)$ of atoms satisfying $\delta(q, l(s))$ such that

- for all universal atoms $(q', \Box, A')$ in $\mathfrak{A}$, there exists a decision $d \in \varDelta^{A'}$ of the agents in A' such that, for all counter decisions $d' \in \varDelta^{A \smallsetminus A'}$, $(q', \tau(s, (d, d'))) \in L$, and
- for all existential atoms $(q', \Diamond, A')$ in $\mathfrak{A}$ and all decisions $d' \in \varDelta^{A \smallsetminus A'}$ of the agents not in A', there exists a counter decision $d \in \varDelta^{A'}$ such that $(q', \tau(s, (d, d'))) \in L$.

A run tree is accepting iff all paths satisfy the Co-Büchi condition that only finitely many positions on the path are labeled with a final state (or rather: with an element of $F \times S$), and a CGS is accepted iff it has an accepting run tree.

The *atoms* of an ACG $\mathcal{A}$ are the elements of the set $atom(\mathcal{A}) \subseteq Q \times (\{\Box, \Diamond\} \times 2^A)$ of atoms that actually occur in some Boolean function $\delta(q, \sigma)$, and the *size* $|Q| + |atom(\mathcal{A})|$ of $\mathcal{A}$ is the sum of the number of its states and atoms.

Theorem 1. *[9] A constructive non-emptiness test of an ACG can be performed in time exponential in the size of the ACG.* □

An automaton is called *universal* if all occurring Boolean functions $\delta(q, \sigma)$ are conjunctions of atoms in $Q \times \{(\Box, \emptyset)\}$.

3 From General to Explicit Models

In this section we show that every model of a specification can be transformed into an *explicit model*, which makes both the truth of each basic subformula in the respective state and a (counter) strategy that witnesses the validity or invalidity of this basic subformulas explicit. This result is exploited in the following section by constructing a small ACG $\mathcal{A}_\varphi$ that accepts the explicit models of φ. Constructing an explicit model from a general model consists of three steps:

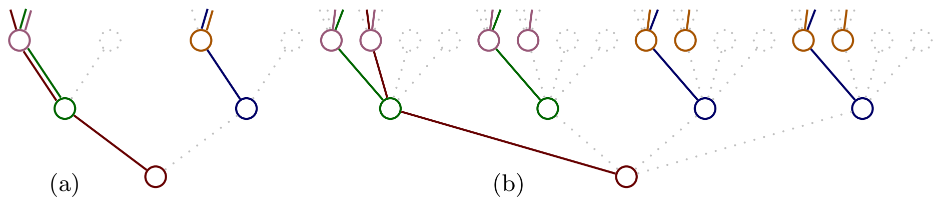

Fig. 2. In the central third step of the transformation of an arbitrary model into an explicit CGT, a CGT is *widened* in order to enable a finite encoding of witness strategies for the (in)validity of basic subformulas in the labels. Figure 2a shows a CGT for a single agent a and a binary set $\Delta = \{left, right\}$, where $\langle\langle a\rangle\rangle\varphi$ holds in every position. The color coding maps a witness strategy for $\langle\langle a\rangle\rangle\varphi$ to every position p – in the single agent case an infinite path rooted in p that satisfies φ. In Figure 2a, the path that always turns left is a witness strategy for the validity of $\langle\langle a\rangle\rangle\varphi$ in the root, indicated by coloring this path and the root of the tree in the same color (red). In general, witness strategies cannot be finitely encoded in the labels of a CGT, because there is no bound on the number of paths a position belongs to. The tree is therefore *widened* by extending Δ to $\Delta' = \{(left, new), (left, cont), (right, new), (right, cont)\}$ (Figure 2b). Witness strategies for the resulting CGT are constructed from witness strategies for the original CGT by turning first to a *new*, and henceforth to a *cont* direction, avoiding the unbounded overlap of witness strategies – for $\langle\langle a\rangle\rangle\varphi$, every position p occurs in at most one witness strategy that does not start in p – allowing for their finite representation.

1. In a first step, we add a fresh atomic proposition b for each basic subformula b of φ, and extend the labeling function such that $b \in l(s) \Leftrightarrow s \in \|b\|_{\mathcal{C}}$.
2. In a second step, we unravel the model obtained in the first step to a tree. Using trees guarantees that no position can be part of infinitely many witnesses. However, the number of witness strategies a position might belong to remains unbounded. (Or: May be linear in the number of its predecessors.)
3. In a final step, we *widen* the tree by adding a single Boolean decision to the set Δ of decisions available to every agent (cf. Figure 2).
   This widening allows us to map arbitrary but fixed witness (counter) strategies from the original tree to witness (counter) strategies in the widened tree such that witnesses for the validity of the same basic subformula b (or its negation $\neg b$) in different states do not overlap. (With the exception of the trivial case that the witness strategy must cover all successors.) This allows us to explicitly encode the witnesses in the widened strategy trees.

From Models to Basic Models. For a given ATL* specification φ, we denote with B_φ the set of its basic subformulas. We call a model $\mathcal{C} = (P \uplus B_\varphi, A, S, s_0, l, \Delta, \tau) \models \varphi$ of an ATL* formula φ *basic* if, for all basic subformulas $b \in B_\varphi$ of φ and all states $s \in S$ of $\mathcal{C}$, $b \in l(s) \Leftrightarrow s \in \|b\|_{\mathcal{C}}$. Since the additional propositions B_φ do not occur in the specification, the following lemma holds trivially:

Lemma 1. *An ATL* formula is satisfiable iff it has a basic model.* □

From Models to Tree Models. We call a CGS $\mathcal{C} = (P, A, S, s_0, l, \Delta, \tau)$ a *concurrent game tree* (CGT) if $S = (\Delta^A)^*$, $s_0 = \varepsilon$, and $\tau(s, d) = s \cdot d$. For a CGS $\mathcal{C} = (P, A, S, s_0, l, \Delta, \tau)$, we call $\mathcal{T}_{\mathcal{C}} = (P, A, (\Delta^A)^*, \varepsilon, l \circ u, \Delta, \tau')$ where

$\tau'(s,d) = s \cdot d$, and where the *unraveling function* $u : (\Delta^A)^* \to S$ is defined recursively by $u(\varepsilon) = s_0$, and $u(s) = s' \Rightarrow u(s \cdot d) = \tau(s', d)$, the *unraveling* of $\mathcal{C}$. We extend u to finite and infinite paths $(u(s_0 s_1 s_2 \ldots) = u(s_0)u(s_1)u(s_2)\ldots)$.

Lemma 2. *A CGS $\mathcal{C}$ is a (basic) model of a specification φ if and only if its unraveling $\mathcal{T}_\mathcal{C}$ is a (basic) model of φ.*

Proof. By induction over the structure of φ, it is easy to prove that $s \in \|\varphi\|_{\mathcal{T}_\mathcal{C}} \Leftrightarrow u(s) \in \|\varphi\|_\mathcal{C}$, and $\pi \in \|\varphi\|^{path}_{\mathcal{T}_\mathcal{C}} \Leftrightarrow u(\pi) \in \|\varphi\|^{path}_\mathcal{C}$. The only non-trivial part in the induction is the transformation of the witness strategies for basic formulas ($\varphi = \langle\!\langle A' \rangle\!\rangle \psi$). However, we can simply use the unraveling function u to transform a witness (counter) strategy $f_{A'}$ or $f^c_{A \smallsetminus A'}$ for $\mathcal{C}$ into a witness (counter) strategy $f'_{A'}$ or ${f^c_{A \smallsetminus A'}}'$, respectively, for $\mathcal{T}_\mathcal{C}$. For this, we fix $f'_{A'}(\pi) = f_{A'}(u(\pi))$ or ${f^c_{A \smallsetminus A'}}'(\pi, d) = f^c_{A \smallsetminus A'}(u(\pi), d)$, respectively. This ensures that $plays(u(s), f_{A'}) = u(plays(s, f'_{A'})) := \{u(\pi) \mid \pi \in plays(s, f'_{A'})\}$, or $plays(u(s), f^c_{A \smallsetminus A'}) = u(plays(s, {f^c_{A \smallsetminus A'}}'))$. Using the induction hypothesis, we get $plays(s, f'_{A'}) \subseteq \|\psi\|^{path}_\mathcal{C}$ or $plays(s, {f^c_{A \smallsetminus A'}}') \subseteq \|\neg\psi\|^{path}_\mathcal{C}$, respectively. □

From Tree Models to Explicit Tree Models. For a CGT $\mathcal{T} = (P, A, (\Delta^A)^*, \varepsilon, l, \Delta, \tau)$, we call the CGT $\mathcal{T}_w = (P, A, ({\Delta'}^A)^*, \varepsilon, l \circ h, \Delta', \tau')$, where $\Delta' = \Delta \times \{new, cont\}$, $h : ({\Delta'}^A)^* \to (\Delta^A)^*$ is a hiding function that hides the $\{new, cont\}$ part of a trace position-wise, and $\tau'(s,d) = s \cdot d$ is the usual transition function of trees, the (Boolean) *widening* of $\mathcal{T}$.

Lemma 3. *A CGT $\mathcal{T}$ is a (basic) model of a specification φ if and only if its (Boolean) widening $\mathcal{T}_w$ is a (basic) model of φ.*

Proof. By induction over the structure of φ. Again, the only non-trivial part is the transformation of the witness strategies for basic formulas ($\varphi = \langle\!\langle A' \rangle\!\rangle \psi$). For this part, we can use the hiding function h to transform a witness strategy $f_{A'}$ in $\mathcal{T}$ into a witness strategy $f'_{A'}$ in its widening $\mathcal{T}_w$ by choosing $f'_{A'}(\pi) = (f_{A'}(h(\pi)), *)$, where $* \in \{new, cont\}$ can be chosen arbitrarily. This ensures $plays(h(s), f_{A'}) = h(plays(s, f'_{A'})) := \{h(\pi) \mid \pi \in plays(s, f'_{A'})\}$. Using the induction hypothesis, we get $plays(s, f'_{A'}) \subseteq \|\psi\|^{path}_\mathcal{C}$. As in the previous lemma, we get the analogous result for the transformation of a witness counter strategy. □

Let, for a basic subformula $B_\varphi \ni b = \langle\!\langle A' \rangle\!\rangle \varphi_b$ of a specification φ, $a(b) = A'$ and $a(\neg b) = A \smallsetminus A'$ denote the set of agents that cooperate to ensure φ_b and the set of their opponents, respectively, and let $E_\varphi = \{(b, new), (b, cont), (\neg b, new), (\neg b, cont) \mid b \in B_\varphi\}$ denote an extended set of subformulas. We call a concurrent game structure $\mathcal{C} = (P \uplus B_\varphi \uplus E_\varphi, A, S, s_0, l, \Delta, \tau)$ *well-formed* if it satisfies the following requirements:

- $\forall s{\in}S.\, b{\notin}l(s) \Rightarrow \forall d{\in}\Delta^{a(b)} \exists d'{\in}\Delta^{a(\neg b)}.\, (\neg b, new){\in}l(\tau(s,(d,d')))$,
- $\forall s{\in}S.\, (\neg b, new){\in}l(s) \Rightarrow \forall d{\in}\Delta^{a(b)} \exists d'{\in}\Delta^{a(\neg b)}.\, (\neg b, cont){\in}l(\tau(s,(d,d')))$,
- $\forall s{\in}S.\, (\neg b, cont){\in}l(s) \Rightarrow \forall d{\in}\Delta^{a(b)} \exists d'{\in}\Delta^{a(\neg b)}.\, (\neg b, cont){\in}l(\tau(s,(d,d')))$,
- $\forall s{\in}S.\, b{\in}l(s) \Rightarrow \exists d{\in}\Delta^{a(b)} \forall d'{\in}\Delta^{a(\neg b)}.\, (b, new){\in}l(\tau(s,(d,d')))$,
- $\forall s{\in}S.\, (b, new){\in}l(s) \Rightarrow \exists d{\in}\Delta^{a(b)} \forall d'{\in}\Delta^{a(\neg b)}.\, (b, cont){\in}l(\tau(s,(d,d')))$, and
- $\forall s{\in}S.\, (b, cont){\in}l(s) \Rightarrow \exists d{\in}\Delta^{a(b)} \forall d'{\in}\Delta^{a(\neg b)}.\, (b, cont){\in}l(\tau(s,(d,d')))$.

For a basic subformula $B_\varphi \ni b = \langle\!\langle a(b)\rangle\!\rangle \varphi_b$ of φ and its negation $\neg b$, we call the set of traces $\mathit{witness}(s,b) = \{ss_1s_2s_3\ldots \in \mathit{path}(s) \mid b \in l(s),\ (b,\mathit{new}) \in l(s_1)$ and $\forall i \geq 2.\, (b,\mathit{cont}) \in l(s_i)\}$ and $\mathit{witness}(s,\neg b) = \{ss_1s_2s_3\ldots \in \mathit{path}(s) \mid b \notin l(s),\ (\neg b,\mathit{new}) \in l(s_1)$ and $\forall i \geq 2.\, (\neg b,\mathit{cont}) \in l(s_i)\}$ the explicit witnesses for b and $\neg b$ in s. $\mathcal{C}$ is called an *explicit model* of φ if the explicit witnesses are contained in the set of paths that satisfy φ_b and $\neg\varphi_b$, respectively. ($\mathit{witness}(s,b) \subseteq \|\varphi_b\|_{\mathcal{C}}^{\mathit{path}}$ and $\mathit{witness}(s,\neg b) \subseteq \|\neg\varphi_b\|_{\mathcal{C}}^{\mathit{path}}$ for all $s \in S$ and $b \in B_\varphi$.) Note that explicit models of φ are in particular basic models of φ.

Lemma 4. *Given a CGT $\mathcal{T}$ that is a basic model of an ATL* formula φ and a set of witness strategies for $\mathcal{T}$, we can construct an explicit model of φ.*

Proof. In the proof of the previous lemma, we showed that the widening $\mathcal{T}_w$ of a basic tree model $\mathcal{T}$ of φ is a basic model of φ. Moreover, we showed that, for the translation of witness (counter) strategies that demonstrate the (in)validity of a subformula $b \in B_\varphi$ of φ in a state s of $\mathcal{T}_w$, *any* extension $* \in \{\mathit{new}, \mathit{cont}\}$ can be chosen. In particular, the agents in $a(b)$ or $a(\neg b)$, respectively, can choose to first pick the *new* extension, and henceforth to pick the extension *cont*. For non-universal specifications, that is, for the case $a(b) \neq \emptyset$ or $a(\neg b) \neq \emptyset$, respectively, this particular choice provides the guarantee that states reachable under the new strategy $f'_{a(b)}$ or counter strategy ${f^c_{a(\neg b)}}'$, respectively, from different states in $\mathcal{T}_w$ are disjoint. ($\forall s_1, t_1 \in ({\Delta'}^A)^*\, \forall i,j > 1.\, s_1s_2s_3\ldots \in \mathit{plays}(s_1, f'_{a(b)}) \wedge t_1t_2t_3\ldots \in \mathit{plays}(t_1, f'_{a(b)}) \wedge s_i = t_j \Rightarrow s_1 = t_1$, and the analogous result for ${f^c_{a(\neg b)}}'$.)

For universal specifications, that is, for the case $a(b) = \emptyset$ or $a(\neg b) = \emptyset$, respectively, the respective player intuitively has no choice, and the (counter) strategy $f'_{a(b)}$ or ${f^c_{a(\neg b)}}'$ is well defined.

In both cases, we mark the positions reachable under $f'_{a(b)}$ in one step from a position s_1 with $b \in l(s_1)$ by (b,new) and positions reachable under ${f^c_{a(\neg b)}}'$ in one step from a position s_1 with $b \notin l(s_1)$ by $(\neg b,\mathit{new})$, and we mark positions reachable in *more* than one step by (b,cont) and $(\neg b,\mathit{cont})$, respectively.

By construction, the resulting CGT $\mathcal{T}_w$ is well-formed, and $b \in l(s) \Rightarrow \mathit{witness}(s,b) = \mathit{plays}(s, f'_{a(b)})$ and $b \notin l(s) \Rightarrow \mathit{witness}(s,\neg b) = \mathit{plays}(s, {f^c_{a(\neg b)}}')$ hold. By Lemma 3, we also get $b \in l(s) \Rightarrow \mathit{plays}(s, f'_{a(b)}) \subseteq \|\varphi_b\|_{\mathcal{C}}^{\mathit{path}}$ and $b \notin l(s) \Rightarrow \mathit{plays}(s, {f^c_{a(\neg b)}}') \subseteq \|\neg\varphi_b\|_{\mathcal{C}}^{\mathit{path}}$. □

Theorem 2. *A specification has an explicit model if and only if it has a model.*

Proof. The 'if' direction is implied by the Lemmata 1–4. For the 'only if' direction, it is obvious that, for a given explicit model $(P \uplus B_\varphi \uplus E_\varphi, A, S, s_0, l, \Delta, \tau)$ of an ATL* formula φ, and for the projection of the labeling function to the atomic propositions ($l'(s){=}l(s){\cap}P$), $(P, A, S, s_0, l', \Delta, \tau)$ is a model of φ. □

4 ATL* Satisfiability is 2EXPTIME-Complete

We exploit the explicit model theorem by constructing an ACG $\mathcal{A}_\varphi$ from an ATL* specification φ that accepts only the explicit models of φ. Testing if a CGS is a model of φ is considerably harder than testing if it is an explicit model. The latter only comprises two simple tests: Checking the well-formedness criterion can be performed by a (safety) ACG with $O(|B_\varphi|)$ states, while, for all basic subformulas $b \in B_\varphi$ of φ, testing if all paths in *witness*(s, b) satisfy the path formula φ_b and if all paths in *witness*$(s, \neg b)$ satisfy the path formula $\neg\varphi_b$ can be performed by a universal ACG that is exponential in φ_b.

Automata that check the (much weaker) model property, on the other hand, need to guarantee consistency of the automaton decisions, which is usually solved by using *deterministic* word automata to represent the single φ_b, leading to an exponentially larger ACG (with parity acceptance condition and a number of colors exponential in the length of φ).

We call a CGS $\mathcal{C}$ *plain* if all states in $\mathcal{C}$ are reachable from the initial state. We can restrict our focus without loss of generality to plain concurrent game structures, because unreachable states have no influence on the model property (nor are they traversed by an automaton).

Lemma 5. *For a specification φ, we can build an ACG $\mathcal{A}_w$ with $O(|E_\varphi|)$ states that accepts a plain CGS $\mathcal{C} = (P \uplus B_\varphi \uplus E_\varphi, A, S, s_0, l, \Delta, \tau)$ iff it is well-formed.*

Proof. We can simply set $\mathcal{A}_w = (\Sigma_w, Q_w, q_0^w, \delta, \emptyset)$ with $\Sigma_w = 2^{B_\varphi \uplus E_\varphi}$ (the atomic propositions P are not interpreted), $Q_w = \{q_0^w\} \uplus E_\varphi$, and

– $\delta(q_0^w, \sigma) = (q_0^w, \Box, \emptyset) \wedge \bigwedge_{b \in \sigma \cap B_\varphi}((b, new), \Box, a(b)) \wedge \bigwedge_{b \in B_\varphi \smallsetminus \sigma}((\neg b, new), \Diamond, a(b))$
$\wedge \bigwedge_{(b,*) \in \sigma \cap E_\varphi}((b, cont), \Box, a(b)) \wedge \bigwedge_{(\neg b,*) \in \sigma \cap E_\varphi}((\neg b, cont), \Diamond, a(b))$, and
– for all $e \in E_\varphi$, $\delta(e, \sigma) = true$ if $e \in \sigma$, and $\delta(e, \sigma) = false$ otherwise.

The $(q_0^w, \Box, \emptyset)$ part of the transition function guarantees that every reachable position in the input CGS is traversed, and the remainder of the transition function simply reflects the well-formedness constraints. $\Box$

Theorem 3. *[4] Given an LTL formula φ, we can build an equivalent universal Co-Büchi word automaton whose size is exponential in the length of φ.* $\Box$

In the context of this paper, the equivalent universal word automaton is read as a universal ACG $\mathcal{U}$ that accepts exactly those words that satisfy the LTL formula. (Words can be viewed as special concurrent game structures with a singleton set of decisions ($|\Delta| = 1$) or an empty set of agents ($A = \emptyset$).)

Let, for a path formula ψ, $\widehat{\psi}$ denote the formula obtained by replacing all occurrences of direct basic subformulas $b \in B_\psi$ by b (read as atomic proposition).

Lemma 6. *For a specification φ and every $B_\varphi \ni b = \langle\langle a(b) \rangle\rangle \varphi_b$ we can build two universal ACGs $\mathcal{A}_b$ and $\mathcal{A}_{\neg b}$ whose size is exponential in the size of $\widehat{\varphi_b}$ and that accept a plain CGS $\mathcal{C} = (P \uplus B_\varphi \uplus E_\varphi, A, S, s_0, l, \Delta, \tau)$ iff witness$(s, b) \subseteq \|\widehat{\varphi_b}\|_{\mathcal{C}}^{path}$ and witness$(s, \neg b) \subseteq \|\neg\widehat{\varphi_b}\|_{\mathcal{C}}^{path}$, respectively, hold true.*

Proof. By Theorem 3 we can translate the LTL formula $\widehat{\varphi_b}$ into an equivalent universal ACG $\mathcal{U}_b = (P \uplus B_\varphi, Q_b, q_0^b, \delta_b, F_b)$ whose size is exponential in the length of $\widehat{\varphi_b}$. From $\mathcal{U}_b$, we infer the universal ACG $\mathcal{A}_b = (P \uplus B_\varphi \uplus E_\varphi, Q_b \times \{new, cont\} \uplus \{q_b\}, q_b, \delta, F_b \times \{cont\})$ with the following transition function:

– $\delta(q_b, \sigma) = (q_b, \Box, \emptyset)$ if $b \notin \sigma$ and
– $\delta(q_b, \sigma) = (q_b, \Box, \emptyset) \wedge \bigwedge_{q \in \delta_b(q_0^b, \sigma)}((q, \mathit{new}), \Box, \emptyset)$ otherwise,
– $\delta((q, \mathit{new}), \sigma) = \mathit{true}$ if $(b, \mathit{new}) \notin \sigma$ and
– $\delta((q, \mathit{new}), \sigma) = \bigwedge_{q' \in \delta_b(q, \sigma)}((q', \mathit{cont}), \Box, \emptyset)$ otherwise, and
– $\delta((q, \mathit{cont}), \sigma) = \mathit{true}$ if $(b, \mathit{cont}) \notin \sigma$ and
– $\delta((q, \mathit{cont}), \sigma) = \bigwedge_{q' \in \delta_b(q, \sigma)}((q', \mathit{cont}), \Box, \emptyset)$ otherwise.

δ again uses the $(q_b, \Box, \emptyset)$ part of the transition function to traverse every reachable position in the input CGS. The assignments $\delta((q, *), \sigma) = \mathit{true}$ ensure that, starting in any reachable state s, only the infinite paths in $\mathit{witness}(s, b)$ are traversed. The remaining transitions reflect the requirement that, for all reachable positions s, all paths in $\mathit{witness}(s, b)$ must satisfy the path formula $\widehat{\varphi_b}$.

$\mathcal{A}_{\neg b}$ can be constructed analogously. □

Theorem 4. *For a given ATL* specification φ, we can construct an ACG $\mathcal{A}_\varphi$ that is exponential in the size of φ and that accepts a plain CGS if and only if it is an explicit model of φ.*

Proof. We build the automaton $\mathcal{A}_\varphi = (2^{P \uplus B_\varphi \uplus E_\varphi}, \{q_0\} \uplus Q_w \uplus \biguplus_{b \in B_\varphi}\{q_b, q_{\neg b}\} \uplus (Q_b \uplus Q_{\neg b}) \times \{\mathit{new}, \mathit{cont}\}, q_0, \delta, \biguplus_{b \in B_\varphi}(F_b \uplus F_{\neg b}) \times \{\mathit{cont}\})$ that consists of the states of the ACG $\mathcal{A}_w$ and, for every basic subformula $b \in B_\varphi$ of φ, of the ACGs $\mathcal{A}_b$ and $\mathcal{A}_{\neg b}$, and a fresh initial state q_0. The transition function for the non-initial states is simply inherited from the respective ACG, and for the initial state we set $\delta(q_0, \sigma) = \mathit{false}$ if σ does not satisfy φ (when read as a Boolean formula over atomic propositions and basic subformulas), and $\delta(q_0, \sigma) = \delta(q_0^w, \sigma) \wedge \bigwedge_{b \in B_\varphi} \delta(q_b, \sigma) \wedge \delta(q_{\neg b}, \sigma)$ otherwise.

The lemmata of this section imply that $\mathcal{A}_\varphi$ is exponential in the size of φ, and accepts a plain CGS if and only if it is an explicit model of φ. □

It is only a small step from the non-emptiness preserving reduction of ATL* to a 2EXPTIME algorithm for ATL* satisfiability checking and synthesis.

Together, Theorems 1, 2 and 4 provide a 2EXPTIME algorithm for a constructive satisfiability test for an ATL* specification. The corresponding hardness result can be inferred from the 2EXPTIME completeness [4] of the satisfiability problem for the syntactic sublogic CTL* (and even for CTL$^+$ [15]) of ATL*.

Corollary 1. *The ATL* satisfiability and synthesis problems are 2EXPTIME-complete.* □

5 Conclusions

We showed that the satisfiability and synthesis problem of ATL* specifications is 2EXPTIME-complete. This result is surprising: For the remaining branching-time temporal logics, the satisfiability problem is at least exponentially harder than the model checking problem [4, 14] (in the size of the specification).

What is more, the suggested reduction indicates that ATL* synthesis may be feasible. The exponential blow-up in the construction of the ACG is the same blow-up that occurs when translating an LTL specification to a nondeterministic word automaton. While this blow-up is unavoidable in principle, it is also known that no blow-up occurs in most practical examples. This gives rise to the assumption that, for most practical ATL* specifications φ, the size of the emptiness equivalent Co-Büchi ACG $\mathcal{A}_\varphi$ will be small. Moreover, $\mathcal{A}_\varphi$ is essentially universal (plus a few simple local constraints), and synthesis procedures for universal Co-Büchi automata have recently seen a rapid development (cf. [16–18]).

ATL* specifications thus seem to be particularly well suited for synthesis: They form one of the rare exceptions of the rule that testing (model-checking) is simpler than constructing a solution.

References

1. Wolper, P.: Synthesis of Communicating Processes from Temporal-Logic Specifications. PhD thesis, Stanford University (1982)
2. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proc. IBM Workshop on Logics of Programs, Springer-Verlag (1981) 52–71
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM **49** (2002) 672–713
4. Emerson, E.A.: Temporal and modal logic. MIT Press (1990)
5. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. Transactions On Programming Languages and Systems **8** (1986) 244–263
6. Kupferman, O., Vardi, M.Y., Wolper, P.: An automata-theoretic approach to branching-time model checking. Journal of the ACM **47** (2000) 312–360
7. Walther, D., Lutz, C., Wolter, F., Wooldridge, M.: Atl satisfiability is indeed exptime-complete. Journal of Logic and Computation **16** (2006) 765–787
8. Kremer, S., Raskin, J.F.: A game-based verification of non-repudiation and fair exchange protocols. Journal of Computer Security **11** (2003) 399–430
9. Schewe, S., Finkbeiner, B.: Satisfiability and finite model property for the alternating-time μ-calculus. In: Proc. CSL, Springer-Verlag (2006) 591–605
10. Even, S., Yacobi, Y.: Relations among public key signature systems. Technical Report 175, Technion, Haifa, Israel (1980)
11. de Alfaro, L., Henzinger, T.A., Majumdar, R.: From verification to control: Dynamic programs for omega-regular objectives. In: Proc. LICS, IEEE Computer Society Press (2001) 279–290
12. van Drimmelen, G.: Satisfiability in alternating-time temporal logic. In: Proc. LICS, IEEE Computer Society Press (2003) 208–217
13. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bull. Soc. Math. Belg. **8** (2001)
14. Kupferman, O., Vardi, M.Y.: Church's problem revisited. The bulletin of Symbolic Logic **5** (1999) 245–263
15. Wilke, T.: CTL$^+$ is exponentially more succinct than CTL. In: Proc. FSTTCS'99, Springer-Verlag (1999) 110–121
16. Kupferman, O., Vardi, M.: Safraless decision procedures. In: Proc. 46th IEEE Symp. on Foundations of Computer Science, Pittsburgh (2005) 531–540
17. Kupferman, O., Piterman, N., Vardi, M.: Safraless compositional synthesis. In: Proc. CAV, Springer-Verlag (2006) 31–44
18. Schewe, S., Finkbeiner, B.: Bounded synthesis. In: Proc. ATVA, Springer Verlag (2007) 474–488