Model-checking Quantitative Alternating-time Temporal Logic on One-counter Game Models

Steen Vester

Technical University of Denmark, Kgs. Lyngby, Denmark

Abstract

We consider quantitative extensions of the alternating-time temporal logics ATL/ATL* called quantitative alternating-time temporal logics (QATL/QATL*) in which the value of a counter can be compared to constants using equality, inequality and modulo constraints. We interpret these logics in one-counter game models which are infinite duration games played on finite control graphs where each transition can increase or decrease the value of an unbounded counter. That is, the state-space of these games are, generally, infinite. We consider the model-checking problem of the logics QATL and QATL* on one-counter game models with VASS semantics for which we develop algorithms and provide matching lower bounds. Our algorithms are based on reductions of the model-checking problems to model-checking games. This approach makes it quite simple for us to deal with extensions of the logical languages as well as the infinite state spaces. The framework generalizes on one hand qualitative problems such as ATL/ATL* model-checking of finite-state systems, model-checking of the branching-time temporal logics CTL and CTL* on one-counter processes and the realizability problem of LTL specifications. On the other hand the model-checking problem for QATL/QATL* generalizes quantitative problems such as the fixed-initial credit problem for energy games (in the case of QATL) and energy parity games (in the case of QATL*). Our results are positive as we show that the generalizations are not too costly with respect to complexity. As a byproduct we obtain new results on the complexity of model-checking CTL* in one-counter processes and show that deciding the winner in one-counter games with LTL objectives is 2ExpSpace-complete.

1 Introduction

The alternating-time temporal logics ATL and ATL* [1] are used to specify temporal properties of systems in which several entities interact. They generalize the widely applied linear-time temporal logic LTL [22] and computation tree logics CTL [9] and CTL* [11] to a multi-agent setting. Indeed, it is possible to specify and reason about what different coalitions of agents can make sure to achieve. The model-checking problem for alternating-time temporal logics subsumes the realizability problem for LTL [23, 24] which is the problem of deciding whether there exists a program satisfying a given LTL specification no matter how the

environment behaves. This is closely related to the synthesis problem which consists of generating a program meeting such a specification. Properties in these logics are inherently qualitative and the model-checking problem for alternating-time temporal logics has primarily been treated in finite-state systems. However, in [7] extensions of ATL and ATL* to the quantitative alternating-time temporal logics QATL and QATL* have been introduced. The purpose is to make the languages capable of expressing quantitative properties of multi-agent scenarios as well as deal with infinite-state systems. These are represented using unbounded counters in addition to a finite set of control states. Naturally, this leads to undecidability in many cases since already deciding the winner in a reachability game on a two-dimensional vector addition system with states (VASS) can already simulate the halting problem of a two-counter machine [6]. In order to regain decidability we focus on the subproblem of a single unbounded counter. This is a significant restriction from the multi-dimensional case, but it still lets us express many interesting properties of infinite-state multi-agent systems. For instance, the model-checking problem includes problems such as energy games [3] and energy parity games [8] in which a system respectively needs to keep an energy level positive and needs to keep an energy level positive while satisfying a parity condition. These are expressible in QATL and QATL* as $\langle\langle \mathtt{Sys} \rangle\rangle \mathbf{G}(r > 0)$ and $\langle\langle \mathtt{Sys} \rangle\rangle (\mathbf{G}(r > 0) \wedge \varphi_{\mathrm{parity}})$ respectively where r is used to denote the current value of the counter. It can be compared to constants using relations in $\{<, \leq, =, \geq, >\}$. $\varphi_{\mathrm{parity}}$ is a parity condition expressed as an LTL formula. It is quite natural to model systems with a resource (e.g. battery level, time, money) using a counter where production and consumption correspond to increasing and decreasing the counter respectively.

Let us give another example of a QATL specification. Consider the game in Figure 1 modelling the interaction between the controller of a vending machine and an environment. The environment controls the rectangular states and the controller controls the circular state. Initially, the environment can insert a coin or request coffee. Upon either input the controller can decrease or increase the balance, dispense coffee or release control to the environment again.
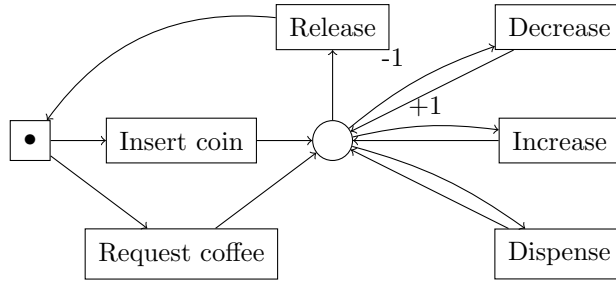

Figure 1: Model of interaction between a vending machine controller and an environment.

Some examples of specifications in QATL* using this model are

- $\langle\langle \{\mathtt{ctrl}\} \rangle\rangle \mathbf{G}(\mathrm{Request} \wedge (r < 3) \rightarrow \mathbf{X}\mathbf{X}\mathrm{Release})$: The controller can make sure that control is released immediately whenever coffee is requested and the balance is less than 3.

- $\langle\!\langle\{\texttt{ctrl}\}\rangle\!\rangle\mathbf{G}(\text{Request} \wedge (r \geq 3) \rightarrow \mathbf{F}\text{Dispense})$: The controller can make sure that whenever coffee is requested and the balance is at least 3 then eventually a cup of coffee is dispensed.

1.1 Contribution

The contribution of this paper is to present algorithms and complexity results for model-checking QATL and QATL* in one-counter game models with one-dimensional VASS semantics, meaning that transitions that would make the counter go below zero are disabled. The complexity is investigated both in terms of whether only edge weights in $\{-1, 0, +1\}$ can be used or if we allow any integer weights encoded in binary. We also distinguish between data complexity and combined complexity. In data complexity, the formula is assumed to be fixed whereas in combined complexity both the formula and the game are parameters. We characterize the complexity of the model-checking problems that arise from these distinctions for both QATL and QATL*. In most of the cases the complexity results are quite satisfying compared with other results from the litterature. As a byproduct we also obtain precise data complexity for model-checking CTL* in one-counter processes (OCPs) and succinct one-counter processes (SOCPs). In addition, we show that the complexity of deciding the winner in a one-counter game with LTL objectives is 2ExpSpace-complete. The complexity results encountered range from PSpace to 2ExpSpace, an overview of the results can be seen in Section 6. The algorithms are based on model-checking games which makes it simple for us to handle the extensions of the logics considered as well as dealing with infinite state spaces and nesting of strategic operators.

1.2 Related work

The realizability problem for LTL was shown to be 2ExpTime-complete in [23, 24]. As this problem is subsumed in QATL* model-checking this gives us an immediate 2ExpTime lower bound for the combined complexity of QATL* model-checking. The results for realizability of LTL specifications are generalized to quantitative objectives in [2] where LTL objectives combined with a mean-payoff objective or an energy objective are considered. However, the semantics in their setting differs from ours in the way the counter value is handled when it gets close to 0. In our setting VASS semantics is used which is not the case in their setting. Our setting is equivalent to one-dimensional VASS games considered in e.g. [6]. Deciding the winner in games played on pushdown processes with parity objectives and LTL objectives were shown to be ExpTime-complete and 3ExpTime-complete in [27] and [19] respectively. Their setting is the same as ours except that in our setting a singleton stack alphabet is used to obtain one-counter games. In [25] it was shown that deciding the winner in one-counter parity games is in PSpace. It follows from [6] that this problem is PSpace-complete since selective zero-reachability in 1-dimensional VASS games is PSpace-hard. The approaches of module checking [18] and in particular pushdown module checking [5] are related to our setting and have given inspiration for our 2ExpSpace-hardness proof of model-checking QATL*. To compare, pushdown module checking of CTL and CTL* are 2ExpTime-complete and 3ExpTime-complete respectively. Our problems generalize several model-

checking problems of branching-time logics in one-counter processes and are related to model-checking in pushdown processes as well. Model-checking of CTL* on pushdown processes has been shown decidable [13], to be in 2EXPTIME [12] and to be 2EXPTIME-hard [4]. On the other hand, model-checking CTL in succinct one-counter processes is EXPSPACE-complete [14]. Other related lines of research includes model-checking of Presburger LTL [10] where counter constraints similar to (and more general than) ours are considered in the linear-time paradigm.

2 Preliminaries

A *one-counter game* (OCG) is a particular kind of finitely representable infinite-state turn-based game. Such a game is represented by a finite game graph where each transition is labelled with an integer value from the set $\{-1, 0, 1\}$ as well as a counter that can hold any non-negative value. The idea is that when a transition labelled v is taken when the counter value is c, the counter value changes to $c + v$. We require that transitions are only applicable when $c + v \geq 0$ since the counter cannot hold a negative value. When $r + v < 0$ we also say that the transition is disabled.

Definition 1. *A one-counter game is a tuple* $\mathcal{G} = (S, \Pi, (S_j)_{j \in \Pi}, R)$ *where*

- S *is a finite set of states*

- Π *is a finite set of players*

- $S = \bigcup_{j \in \Pi} S_j$ *and* $S_i \cap S_j = \emptyset$ *for all* $i, j \in \Pi$ *such that* $i \neq j$

- $R \subseteq S \times \{-1, 0, 1\} \times S$ *is the transition relation*

An OCG is played by placing a token in an initial state s_0 and then moving the token between states for an infinite number of rounds. The transitions must respect the transition relation and the intuition is that for each $j \in \Pi$ player j controls the successor state when the token is placed on a state in S_j. At a given point in the game, the current counter value is given by the sum of the initial value $v_0 \in \mathbb{N}$ and all the edge weights encountered so far. If a transition would make the current counter value decrease below 0 then the transition is disabled. More formally, an element $c \in S \times \mathbb{N}$ is called a configuration of the game. We denote by $(S \times \mathbb{N})^*$, $(S \times \mathbb{N})^+$ and $(S \times \mathbb{N})^\omega$ the set of finite sequences, the set of non-empty finite sequences and the set of infinite sequences of configurations respectively. For a sequence $\rho = c_0 c_1 ...$ we define $\rho_i = c_i$, $\rho_{\leq i} = c_0 ... c_i$ and $\rho_{\geq i} = c_i c_{i+1} ...$. When ρ is finite, i.e. $\rho = c_0 ... c_\ell$ we write $\text{last}(\rho) = c_\ell$ and $|\rho| = \ell$. A play is a maximal sequence $\rho = (s_0, v_0)(s_1, v_1)...$ of configurations such that for all $i \geq 0$ we have $(s_i, v_{i+1} - v_i, s_{i+1}) \in R$ and $v_i \geq 0$. A history is a proper prefix of a play. The set of plays and histories in an OCG $\mathcal{G}$ are denoted by $\text{Play}_\mathcal{G}$ and $\text{Hist}_\mathcal{G}$ respectively (the subscript may be omitted when it is clear from the context). The set of plays and histories with initial configuration c_0 are denoted $\text{Play}_\mathcal{G}(c_0)$ and $\text{Hist}_\mathcal{G}(c_0)$ respectively. A strategy for player $j \in \Pi$ in $\mathcal{G}$ is a partial function $\sigma : \text{Hist}_\mathcal{G} \to S \times \mathbb{N}$ defined for all histories $h = (s_0, v_0)...(s_\ell, v_\ell) \in \text{Hist}_\mathcal{G}$ such that $s_\ell \in S_j$ with the requirement that if $\sigma(h) = (s, v)$ then $(s_\ell, v - v_\ell, s) \in R$. A play (resp. history)

$\rho = c_0 c_1 ...$ (resp. $\rho = c_0 ... c_\ell$) is compatible with a strategy σ_j for player $j \in \Pi$ if $\sigma_j(\rho_{\leq i}) = \rho_{i+1}$ for all $i \geq 0$ (resp. $0 \leq i < \ell$) such that $\rho_i \in S_j \times \mathbb{N}$. We denote by $\mathrm{Strat}_{\mathcal{G}}^j$ the set of strategies of player j in $\mathcal{G}$. For a coalition $A \subseteq \Pi$ of players a collective strategy $\sigma = (\sigma_j)_{j \in A}$ is a tuple of strategies, one for each player in A. We denote by $\mathrm{Strat}_{\mathcal{G}}^A$ the set of collective strategies of coalition A. For an initial configuration c_0 and collective strategy $\sigma = (\sigma_j)_{j \in A}$ of coalition A we denote by $\mathrm{Play}(c_0, \sigma)$ the set of plays with initial configuration c_0 that are compatible with σ_j for every $j \in A$.

We extend one-counter games such that arbitrary integer weights are allowed and such that transitions are still disabled if they would make the counter go below zero. Such games are called succinct one-counter games (SOCGs). We suppose that weights are given in binary. The special cases of OCGs and SOCGs where Π is a singleton are called one-counter processes (OCPs) and succinct one-counter processes (SOCPs) respectively. A game model $\mathcal{M} = (\mathcal{G}, \mathrm{AP}, L)$ consists of a (one-counter or succinct one-counter) game $\mathcal{G}$, a finite set AP of atomic proposition symbols and a labelling $L : S \mapsto 2^{\mathrm{AP}}$ of the states S of the game $\mathcal{G}$ with atomic propositions. We abbreviate one-counter game models and succinct one-counter game models by OCGM and SOCGM respectively.

By a one-counter parity game we mean the particular kind of one-counter game model where there are two players I and II and the set of propositions is a finite subset of the natural numbers, called colors. Further, every control state is labelled with exactly one color. In such a game, player I wins if the least color occuring infinitely often is even. Otherwise player II wins. We assume that the counter value is 0 initially and that there is a designated initial state in a one-counter parity game. It was shown in [25] that the winner can be determined in a one-counter parity game in polynomial space by a reduction to the emptiness problem for alternating two-way parity automata [26].

Proposition 2. *Determining the winner in one-counter parity games is in* PSpace.

3 Quantitative Alternating-time temporal logic

We consider fragments of the quantitative alternating-time temporal logics QATL and QATL* introduced in [7] interpreted over one-counter game models. The logics extend the standard ATL and ATL* [1] with atomic formulas of the form $r \bowtie c$ where $c \in \mathbb{Z}$ and $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ with $k \in \mathbb{N}$. They are interpreted in configurations of the game such that $r \leq 5$ is true if the current value of the counter is at most 5 and $r = 0$ is true if the current value of the counter is 0. $r \equiv_4 3$ means that the current value of the counter is equivalent to 3 modulo 4. More formally, the formulas of QATL* are defined with respect to a set AP of proposition symbols and a finite set Π of agents. They are constructed using the following grammar

$$\Phi ::= p \mid r \bowtie c \mid \neg \Phi_1 \mid \Phi_1 \vee \Phi_2 \mid \mathbf{X}\Phi_1 \mid \Phi_1 \mathbf{U}\Phi_2 \mid \langle\!\langle A \rangle\!\rangle \Phi_1$$

where $p \in \mathrm{AP}$, $c \in \mathbb{Z}$, $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ with $k \in \mathbb{N}$, $A \subseteq \Pi$ and Φ_1, Φ_2 are QATL* formulas. We define the syntactic fragment QATL of QATL* by the grammar

$$\varphi ::= p \mid r \bowtie c \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid \langle\!\langle A \rangle\!\rangle \mathbf{X}\varphi_1 \mid \langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1 \mid \langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U}\varphi_2$$

where $p \in \mathrm{AP}$, $c \in \mathbb{Z}$, $\bowtie \in \{\leq, <, =, >, \geq, \equiv_k\}$ with $k \in \mathbb{N}$, $A \subseteq \Pi$ and φ_1, φ_2 are QATL formulas. Formulas of the form $r \bowtie c$ are called counter constraints.

We interpret formulas of QATL and QATL* in OCGMs. In standard ATL* we have state formulas and path formulas which are interpreted in states and plays respectively. For QATL and QATL* we also need the value of the counter to interpret state formulas. Note that the value of the counter is already present in a play. The semantics of a formula is defined with respect to a given OCGM $\mathcal{M} = (S, \Pi, (S_j)_{j\in\Pi}, R, \mathrm{AP}, L)$ inductively on the structure of the formula. For all states $s \in S$, plays $\rho \in \mathrm{Play}_{\mathcal{M}}$, $p \in \mathrm{AP}$, $c, i \in \mathbb{Z}$, $A \subseteq \Pi$, QATL* state formulas Φ_1, Φ_2 and QATL* path formulas Ψ_1, Ψ_2 let the satisfaction relation $\models$ be given by

$$
\begin{array}{llll}
\mathcal{M}, s, i & \models p & \text{iff} & p \in L(s) \\
\mathcal{M}, s, i & \models r \bowtie c & \text{iff} & i \bowtie c \\
\mathcal{M}, s, i & \models \neg\Phi_1 & \text{iff} & \mathcal{M}, s, i \not\models \Phi_1 \\
\mathcal{M}, s, i & \models \Phi_1 \vee \Phi_2 & \text{iff} & \mathcal{M}, s, i \models \Phi_1 \text{ or } \mathcal{M}, s, i \models \Phi_2 \\
\mathcal{M}, s, i & \models \langle\!\langle A \rangle\!\rangle \Psi_1 & \text{iff} & \exists\sigma \in \mathrm{Strat}_{\mathcal{M}}^{A}.\forall\pi \in \mathrm{Play}_{\mathcal{M}}((s, i), \sigma).\mathcal{M}, \pi \models \Psi_1 \\
\\
\mathcal{M}, \rho & \models \Phi_1 & \text{iff} & \mathcal{M}, \rho_0 \models \Phi_1 \\
\mathcal{M}, \rho & \models \neg\Psi_1 & \text{iff} & \mathcal{M}, \rho \not\models \Psi_1 \\
\mathcal{M}, \rho & \models \Psi_1 \vee \Psi_2 & \text{iff} & \mathcal{M}, \rho \models \Psi_1 \text{ or } \mathcal{M}, \rho \models \Psi_2 \\
\mathcal{M}, \rho & \models \mathbf{X}\Psi_1 & \text{iff} & \mathcal{M}, \rho_{\geq 1} \models \Psi_1 \\
\mathcal{M}, \rho & \models \Psi_1 \mathbf{U}\Psi_2 & \text{iff} & \exists k \geq 0.\mathcal{M}, \rho_{\geq k} \models \Psi_2 \text{ and } \forall 0 \leq i < k.\mathcal{M}, \rho_{\geq i} \models \Psi_1
\end{array}
$$

The definition of the semantics is extended in the natural way to SOCGMs.

In this paper we focus on the model-checking problem. That is to decide, given an OCGM/SOCGM $\mathcal{M}$, a state s in $\mathcal{M}$, a natural number i and a QATL/QATL* formula φ whether $\mathcal{M}, s, i \models \varphi$. When doing model-checking we assume that states are only labelled with atomic propositions that occur in the formula φ as well as the special propositions $\top$ and $\bot$ that are true in all states and false in all states respectively. This is done to ensure that the input is finite. When measuring the complexity of the model-checking problem we distinguish between data complexity and combined complexity. For data complexity, the formula φ is assumed to be fixed and thus, the complexity only depends on the model. For combined complexity both the formula and game are assumed to be parameters. When model-checking OCGMs, the initial counter value i is assumed to be input in unary and for SOCGMs, the initial counter value i is assumed to be input in binary.

4   Model-checking QATL

When model-checking ATL and ATL* in finite-state systems, the standard approach is to process the state subformulas from the innermost to the outermost, at each step labelling all states where the subformula is true. This approach does not work directly in our setting since we have an infinite number of configurations. We therefore take a different route and develop a model-checking game

in which we can avoid explicitly labelling the configurations in which a subformula is true. This approach also allows us to handle the counter constraints in a natural way.

4.1 A model-checking game for QATL

We convert the model-checking problem asking whether $\mathcal{M}, s_0, i \models \varphi$ for a QATL formula φ in a configuration (s_0, i) of an OCGM $\mathcal{M} = (S, \Pi, (S_j)_{j \in \Pi}, R, \text{AP}, L)$ to a model-checking game $\mathcal{G}_{\mathcal{M}, s_0, i}(\varphi)$ between two players Verifier and Falsifier that are trying to respectively verify and falsify the formula. The construction is done so Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M}, s_0, i}(\varphi)$ if and only if $\mathcal{M}, s_0, i \models \varphi$. The model-checking game can be constructed in polynomial time and is an OCG with a parity winning condition. According to Proposition 2 determining the winner in such a game can be done in PSPACE.

The construction is done inductively on the structure of φ. For a given QATL formula, a given OCGM $\mathcal{M}$ and a given state s in $\mathcal{M}$ we define a characteristic OCG $\mathcal{G}_{\mathcal{M}, s}(\varphi)$. Note that the initial counter value is not present in the construction yet. There are a number of different cases to consider. We start with the base cases where φ is either a proposition p or a formula of the form $r \bowtie c$ and then move on to the inductive cases. The circle states are controlled by Verifier and square states are controlled by Falsifier. Verifier wins the game if the least color that appears infinitely often during the play is even, otherwise Falsifier wins the game. The states are labelled with colors whereas edges are labelled with counter updates.

$\mathcal{G}_{\mathcal{M}, s}(p)$: There are two cases. When $p \in L(s)$ and when $p \notin L(s)$. The two resulting games are illustrated in Figure 2 to the left and right respectively.

Figure 2: $\mathcal{G}_{\mathcal{M}, s}(p)$. To the left is the case where $p \in L(s)$ and to the right is the case where $p \notin L(s)$

$\mathcal{G}_{\mathcal{M}, s}(r \bowtie c)$: Using negation and conjunction we can define $(r = c) \equiv (r \leq c \wedge \neg(r < c))$, $(r > c) \equiv (\neg(r \leq c))$ and $(r \geq c) \equiv (\neg(r < c))$ and therefore only need to construct games for the cases $r < c$, $r \leq c$ and $r \equiv_k c$. The three cases are shown in Figure 3.

$\mathcal{G}_{\mathcal{M}, s}(\varphi_1 \vee \varphi_2)$: The game is shown in Figure 4.

$\mathcal{G}_{\mathcal{M}, s}(\neg \varphi_1)$: The game is constructed from $\mathcal{G}_{\mathcal{M}, s}(\varphi_1)$ by interchanging circle states and square states and either adding or subtracting 1 to/from all colors.

$\mathcal{G}_{\mathcal{M}, s}(\langle\!\langle A \rangle\!\rangle \mathbf{X} \varphi_1)$: Let $R(s) = \{(s, v, s') \in R\} = \{(s, v_1, s_1), ..., (s, v_m, s_m)\}$. There are two cases to consider. One when $s \in S_j$ for some $j \in A$ and one when $s \notin S_j$ for all $j \in A$. Both are illustrated in Figure 5.

$\mathcal{G}_{\mathcal{M}, s}(\langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi_1)$: In this case we let $\mathcal{G}_{\mathcal{M}, s}(\langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi_1)$ have the same structure as $\mathcal{M}$, but with a few differences. Verifier controls all states that are in S_j for some $j \in A$ and Falsifier controls the other states. Further, for each transition $t = (s', v, s'') \in R$ we put an intermediate state s_t controlled by Falsifier between s' and s''. When the player controlling s' chooses to take the transition t the play is taken to the intermediate state s_t from which Falsifier can either choose to continue to s'' or to go to $\mathcal{G}_{\mathcal{M}, s''}(\varphi_1)$. Every state in $\mathcal{G}_{\mathcal{M}, s}(\langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi_1)$
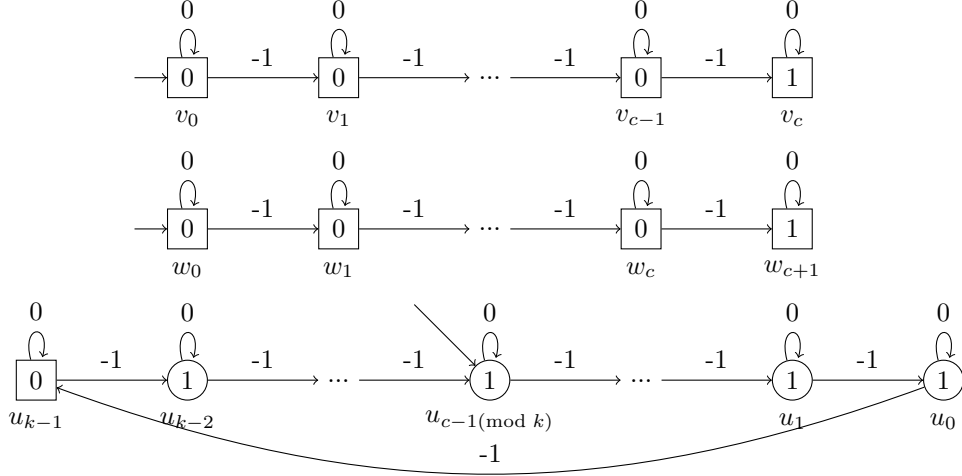
Figure 3: $\mathcal{G}_{\mathcal{M},s}(r < c)$ on top, $\mathcal{G}_{\mathcal{M},s}(r \leq c)$ in the middle and $\mathcal{G}_{\mathcal{M},s}(r \equiv_k c)$ at the bottom.
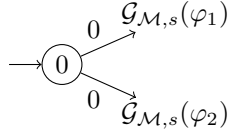

Figure 4: $\mathcal{G}_{\mathcal{M},s}(\varphi_1 \vee \varphi_2)$

which is not part of $\mathcal{G}_{\mathcal{M},s''}(\varphi_1)$ has the color 0. It is illustrated in Figure 6. Diamond states are states that can either be Verifier states or Falsifier states. The intuition is that Falsifier can challenge and claim that φ_1 is not true in the current configuration. If he does so, Verifier must be able show that it is in fact true in order to win.

$\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U} \varphi_2)$: The game is constructed similarly to the case of $\langle\!\langle A \rangle\!\rangle \mathbf{G}$. The differences are that every state is colored by 1 and for each transition $t = (s', v, s'') \in R$ we add two intermediate states s_t and s'_t controlled by Verifier and Falsifier respectively with transitions to $\mathcal{G}_{\mathcal{M},s''}(\varphi_2)$ and $\mathcal{G}_{\mathcal{M},s''}(\varphi_1)$ respectively. The situation is illustrated in Figure 7. The intuition is similar, but in this case Verifier loses unless he can claim φ_2 is true at some point (and subsequently show that this is in fact the case). In addition φ_1 cannot become false before this point, because then Falsifier can claim that φ_1 is false and win.

Finally, we define the game $\mathcal{G}_{\mathcal{M},s,i}(\varphi)$ from $\mathcal{G}_{\mathcal{M},s}(\varphi)$ and a natural number $i \in \mathbb{N}$ as illustrated in Figure 8. Intuitively, this construction is performed to set the initial value of the counter to i.

It is now possible to prove the following result by induction on the structure of the QATL formula φ, giving us a reduction from the model-checking problem to deciding the winner in a one-counter parity game.

Figure 5: $\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A\rangle\!\rangle\mathbf{X}\varphi_1)$. The case on the left is when $s \in S_j$ for some $j \in A$ and the case on the right is when $s \notin S_j$ for all $j \in A$
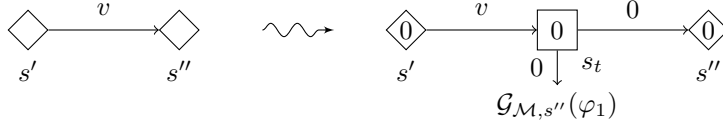

Figure 6: $\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A\rangle\!\rangle\mathbf{G}\varphi_1)$ is obtained by updating each transition in $\mathcal{M}$ as shown in the figure.

Proposition 3. *For every OCGM $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and $\varphi \in$ QATL*

$$\mathcal{M}, s, i \models \varphi \text{ if and only if Verifier has a winning strategy in } \mathcal{G}_{\mathcal{M},s,i}(\varphi)$$

4.2   Complexity

In [6] the selective zero-reachability problem for games on 1-dimensional vector addition systems with states was shown to be PSPACE-complete. This problem consists of model-checking the fixed QATL formula $\langle\!\langle\{I\}\rangle\!\rangle\mathbf{F}(r = 0 \wedge p)$ in a 2-player OCGM where I is one of the players. The hardness is shown by a reduction from the emptiness problem of 1-letter alternating finite automata which is PSPACE-complete [16]. Thus, the data complexity of model-checking QATL in OCGMs is PSPACE-hard. As a consequence of Proposition 3 and Proposition 2 this lower bound is tight since we can transform the model-checking problem of QATL to deciding the winner in an OCG with a parity condition that has polynomial size. Thus, model-checking can be performed in polynomial space.

Theorem 4. *The combined complexity and data complexity of model-checking* QATL *OCGMs are both* PSPACE-*complete*

In [14] it was shown that the data complexity of model-checking CTL in SOCPs is EXPSPACE-complete even for a fixed (but rather complicated) formula. Since this problem is subsumed by the model-checking problem of QATL in SOCGMs we have the same lower bound for the data complexity of model-checking QATL in SOCGMs. It can be shown that this bound is tight as follows. We can create a model-checking game for QATL in SOCGMs in the same way as for OCGMs and obtain a model-checking game which is an SOCG with a parity winning condition. This can be transformed into an OCG with a parity winning condition that is exponentially larger. It is done by replacing each transition with weight v with a path that has v transitions and adding small gadgets to make sure that a player loses if he tries to take a transition with value $-w$ for
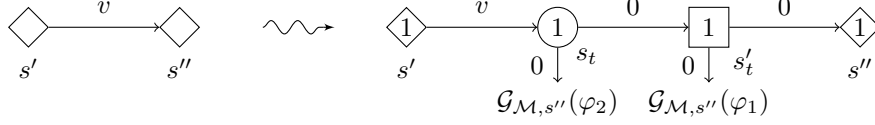
Figure 7: $\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U} \varphi_2)$ is obtained by updating each transition in $\mathcal{M}$ as shown in the figure.
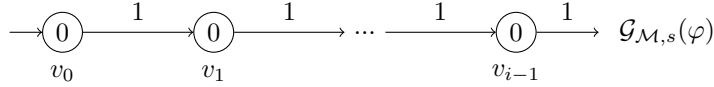

Figure 8: $\mathcal{G}_{\mathcal{M},s,i}(\varphi)$ is obtained by increasing the counter value to i initially.

$w \in \mathbb{N}$ when the current counter value is less than w. The exponential blowup is due to the weights being input in binary. We can then apply Proposition 2 and solve this game in exponential space. Thus, we have the following.

Theorem 5. *The combined complexity and data complexity of model-checking* QATL *in SOCGMs are both* ExpSpace-*complete.*

These results are quite positive. Indeed, in OCGMs reachability games are already PSpace-complete [6]. Considering that in QATL we have nesting of strategic operators, eventuality operators, safety operators and comparison of counter values with constants it is very positive that we stay in the same complexity class. For SOCGMs CTL model-checking is already ExpSpace-complete [14] which means that we can add several players as well as counter constraints without leaving ExpSpace.

5 Model-checking QATL*

As for model-checking of QATL we rely on the approach of a model-checking game when model-checking QATL*. However, due to the extended possibilities of nesting we do not handle temporal operators directly as we did for formulas of the form $\langle\!\langle A \rangle\!\rangle \varphi \mathbf{U} \psi$, $\langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi$ and $\langle\!\langle A \rangle\!\rangle \mathbf{X} \varphi$. Instead, we resort to a translation of LTL formulas into deterministic parity automata (DPA) which is combined with the model-checking game approach. This gives us model-checking games which are one-counter parity games as for QATL, but with doubly exponential size in the input formula due to the translation from LTL formulas to DPAs.

5.1 Adjusting the model-checking game to QATL*

Let $\mathcal{M} = (S, \Pi, (S_j)_{j \in \Pi}, R, AP, L)$ be an OCGM, $s_0 \in S$, $i \in \mathbb{N}$ and φ be a QATL* state formula. The algorithm to decide whether $\mathcal{M}, s_0, i \models \varphi$ follows along the same lines as our algorithm for QATL. That is, we construct a model-checking game $\mathcal{G}_{\mathcal{M},s_0,i}(\varphi)$ between two players Verifier and Falsifier that try to verify and falsify the formula respectively. Then Verifier has a winning strategy

in $\mathcal{G}_{\mathcal{M},s_0,i}(\varphi)$ if and only if $\mathcal{M}, s_0, i \models \varphi$. The construction is done inductively on the structure of φ. For each state $s \in S$ and state formula φ we define a characteristic OCG $\mathcal{G}_{\mathcal{M},s}(\varphi)$. For formulas of the form $p, r \bowtie c, \neg\varphi_1$ and $\varphi_1 \vee \varphi_2$ the construction is as for QATL assuming in the inductive cases that $\mathcal{G}_{\mathcal{M},s}(\varphi_1)$ and $\mathcal{G}_{\mathcal{M},s}(\varphi_2)$ have already been defined.

The interesting case is $\varphi = \langle\!\langle A \rangle\!\rangle \varphi_1$. Here, let $\psi_1, ..., \psi_m$ be the outermost proper state subformulas of φ_1. Let $P = \{p_1, ..., p_m\}$ be fresh propositions and let $f(\varphi_1) = \varphi_1[\psi_1 \mapsto p_1, ..., \psi_m \mapsto p_m]$ be the formula obtained from φ_1 by replacing the outermost proper state subformulas with the corresponding fresh propositions. Let $\mathrm{AP}' = \mathrm{AP} \cup P$. Now, $f(\varphi_1)$ is an LTL formula over AP'. We can therefore construct a deterministic parity automaton (DPA) $\mathcal{A}_{f(\varphi_1)}$ with input alphabet $2^{\mathrm{AP}'}$ such that the language $L(\mathcal{A}_{f(\varphi_1)})$ of the automaton is exactly the set of linear models of $f(\varphi_1)$. The number of states of the DPA can be bounded by $O((2^{n \cdot 2^n})(2^n)!)$ and the number of colors by $O(2 \cdot 2^n) = O(2^{n+1})$ where n is the size of the formula $f(\varphi_1)$. These bounds are obtained by using the fact that a non-deterministic Büchi automaton (NBA) $\mathcal{B}_{f(\varphi_1)}$ with $O(2^n)$ states and $L(\mathcal{B}_{f(\varphi_1)}) = \mathrm{Traces}(f(\varphi_1))$ can be constructed [28]. From this, a DPA accepting the same language can be constructed using a technique from [21] which translates an NBA with m states to a DPA with $2m^m \cdot m!$ states and 2m colors.

The game $\mathcal{G}_{\mathcal{M},s}(\varphi)$ is now constructed with the same structure as $\mathcal{M}$, where Verifier controls the states for players in A and Falsifier controls the states for players in $\Pi \setminus A$. However, we need to deal with truth values of the formulas $\psi_1, ..., \psi_m$ which can in general not be labelled to states in $\mathcal{M}$ since they depend both on the current state and counter value. Therefore we change the structure to obtain $\mathcal{G}_{\mathcal{M},s}(\varphi)$ as follows. For each state s and t with $(s,t) \in R$ we embed a module as shown in Figure 9. Here, $2^{\mathrm{AP}'} = \{\Phi_0, ..., \Phi_\ell\}$ and for each $0 \leq j \leq \ell$ we let $\{\psi_{j0}, ..., \psi_{jk_j}\} = \{\psi_i \mid p_i \in \Phi_j\} \cup \{\neg\psi_i \mid p_i \notin \Phi_j\}$.
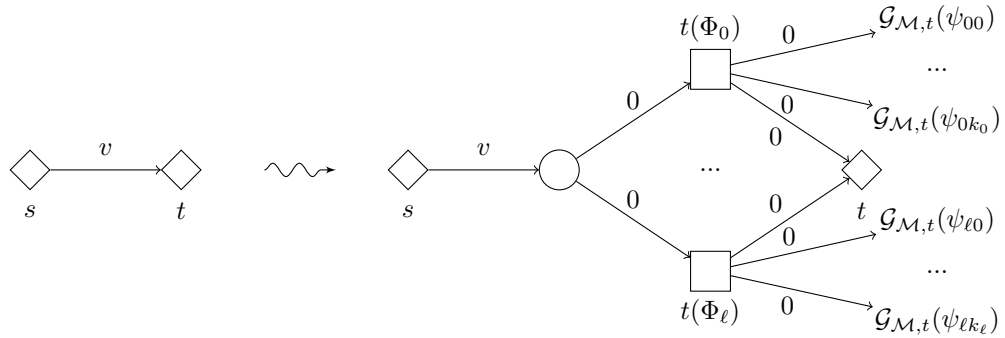

Figure 9: $\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A \rangle\!\rangle \varphi)$ is obtained by updating each transition as shown in the figure.

The idea is that when a transition is taken from (s, w) to $(t, w + v)$, Verifier must specify which of the propositions $p_1, ..., p_m$ are true in $(t, w+v)$, this is done by picking one of the subsets Φ_j (which is the set of propositions that are true in state $t(\Phi_j)$). Then, to make sure that Verifier does not cheat, Falsifier has the opportunity to challenge any of the truth values of the propositions specified by

Verifier. If Falsifier challenges, the play never returns again. Thus, if Falsifier challenges incorrectly, Verifier can make sure to win the game. However, if Falsifier challenges correctly then Falsifier can be sure to win the game. If Verifier has a winning strategy, then it consists in choosing the correct values of the propositions at each step. If Verifier does choose correctly and Falsifier never challenges, the winner of the game should be determined based on whether the LTL property specified by $f(\varphi_1)$ is satisfied during the play. We handle this by labelling $t(\Phi_j)$ with the propositions in Φ_j. Further, since every step of the game is divided into three steps (the original step, the specification by Verifier and the challenge opportunity for Falsifier) we alter the deterministc automaton $\mathcal{A}_{f(\varphi_1)}$ such that it only takes a transition every third step. This simply increases its size by a factor 3. We then perform a product of the game with the updated parity automaton to obtain the parity game $\mathcal{G}_{\mathcal{M},s}(\langle\!\langle A \rangle\!\rangle \varphi_1)$. It is important to note that the product with the automaton is not performed on the challenge modules (which are already colored), but only with states in the main module. This keeps the size of the game double-exponential in the size of the formula. We now have the following.

Proposition 6. *For every OCGM $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and state formula $\varphi \in \text{QATL}^*$*

$$\mathcal{M}, s, i \models \varphi \text{ if and only if Verifier has a winning strategy in } \mathcal{G}_{\mathcal{M},s,i}(\varphi)$$

Proof. Due to space limitations, we only provide a sketch of the proof with the main ideas. The proof is done by induction on the structure of φ. The base cases as well as boolean combinations are omitted since they work as for QATL. The interesting case is $\varphi = \langle\!\langle A \rangle\!\rangle \varphi_1$.

Suppose first that $\mathcal{M}, s, i \models \langle\!\langle A \rangle\!\rangle \varphi_1$. Then coalition A has a winning strategy σ in $\mathcal{M}$. From this, we generate a strategy σ' for Verifier in $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \varphi_1)$ that consists in never cheating when specifying values of atomic formulas and choosing transitions according to what σ would have done in $\mathcal{M}$. Then, if Falsifier challenges at some point, Verifier can be sure to win by the induction hypothesis since he never cheats. If Falsifier never challenges (or, until he challenges), Verifier simply mimics the collective winning strategy σ of coalition A in $\mathcal{M}$ from (s, i). This ensures that he wins in the parity game due to the definition of the parity condition from the parity automaton corresponding to $f(\varphi_1)$.

Suppose on the other hand that Verifier has a winning strategy σ in $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \varphi_1)$. Then σ never cheats when specifying values of propositions, because then Falsifier could win according to the induction hypothesis. Define a strategy σ' for coalition A in $\mathcal{M}$ that plays like σ in the part of $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \varphi_1)$ where no challenge has occured. σ' is winning for A with condition φ_1 in $\mathcal{M}$ due to the definition of $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \varphi_1)$ using the automaton $\mathcal{A}_{f(\varphi_1)}$. $\square$

5.2 Complexity

The size of the model-checking game is doubly-exponential in the size of the formula. Therefore, it can be solved in doubly-exponential space because it is a one-counter parity game using Proposition 2. Actually, this is the case for both OCGMs and SOCGMs. Indeed, we extend the technique to SOCGMs as we did in the case of QATL. However, with respect to complexity, the blowup caused

by the binary representation of edge weights only matters when the formula is fixed since the game is already doubly-exponential when the input formula is a parameter. Thus, for QATL* we can do model-checking in doubly-exponential space whereas for a fixed formula it is in ExpSpace for SOCGMs and PSpace for OCGMs.

For combined complexity we can show that 2ExpSpace is a tight lower bound by a reduction from the word acceptance problem of a doubly-exponential space Turing machine. The reduction reuses ideas from [16], [17] and [5]. The proof is in Appendix B. For a fixed formula we get tight lower bounds immediately from the results on QATL.

Theorem 7. *The combined complexity of model-checking* QATL* *is* 2ExpSpace-*complete for both OCGMs and SOCGMs. The data complexity of model-checking* QATL* *is* PSpace-*complete for OCGMs and* ExpSpace-*complete for SOCGMs.*

Since we have an ExpSpace lower bound for data complexity of CTL model-checking in SOCPs [14] and a PSpace lower bound for data complexity of CTL model-checking in OCPs [15] we get the following results for data complexity of model-checking CTL* in OCPs.

Corollary 8. *The data complexity of model-checking* CTL* *in OCPs and SOCPs are* PSpace-*complete and* ExpSpace-*complete respectively.*

Since our lower bound is for formulas of the form $\langle\!\langle\{I\}\rangle\!\rangle\varphi$ where φ is an LTL formula and I is a player we also have the following.

Corollary 9. *Deciding the winner in two-player OCGs and SOCGs with* LTL *objectives are both* 2ExpSpace-*complete.*

6 Concluding remarks

We have characterized the complexity of the quantitative alternating-time temporal logics QATL and QATL* with respect to the format of edge weights as well as whether the input formula is fixed or not. The results are collected in Table 1. Note that all complexity results on QATL and QATL* hold for ATL and ATL* as well since no counter constraints are used in the proofs of the lower bounds. As a byproduct we have also obtained results for CTL* model-checking on OCPs. These, along with CTL model-checking results on OCPs and SOCPs from the litterature, are included as a comparison.

Given that one-counter reachability games are already PSpace-complete [6] it is very positive that we can extend to QATL model-checking and even to model-checking of fixed QATL* formulas without leaving PSpace. Model-checking CTL in SOCPs is already ExpSpace-complete [14] so it is also very positive that we can extend this to model-checking of QATL and fixed formulas of QATL* in succinct one-counter games. Finally, the 2ExpSpace-completeness results are not too unexpteced compared to the known 2ExpTime lower bound from the synthesis of LTL [23] and 3ExpTime-completeness of pushdown games with LTL objectives [19]. However, though we restrict to a unary stack alphabet compared to pushdown games, we do have counter constraints and nesting of strategic operators.

Finally, the model-checking game approach has turned out to be quite flexible with respect to enriching the alternating-time temporal logics with counter

Table 1: Complexity results of model-checking. Results for QATL and QATL* are on OCGMs and SOCGMs whereas results for CTL and CTL* are for OCPs and SOCPs respectively

	Non-succinct		Succinct	
	Data	Combined	Data	Combined
QATL	PSPACE-c	PSPACE-c	EXPSPACE-c	EXPSPACE-c
QATL*	PSPACE-c	2EXPSPACE-c	EXPSPACE-c	2EXPSPACE-c
CTL	PSPACE-c [15]	PSPACE-c [15]	EXPSPACE-c [14]	EXPSPACE-c [14]
CTL*	PSPACE-c	In 2EXPTIME [12]	EXPSPACE-c	In 2EXPSPACE

constraints. This is also the case when dealing with infinite state-spaces in which labelling of states with formulas that are true is not so straightforward. In addition, it has given us optimal complexity for most of the problems considered. We leave the combined complexity of CTL* model-checking open.

A Full proof of Proposition 3

Proposition 3. *For every OCGM $\mathcal{M}$, state s in $\mathcal{M}$, $i \in \mathbb{N}$ and $\varphi \in$ QATL*

$\mathcal{M}, s, i \models \varphi$ if and only if Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi)$

Proof. The proof is done by induction on the structure of φ. First, we consider the base cases.

$\varphi = p$: In this case Verifier has a winning strategy if and only if $p \in L(s)$ if and only if $\mathcal{M}, s, i \models p$.

$\varphi = (r < c)$: In this case the counter is initially increased to i after i steps of the game. Then, Falsifier can win exactly if he can decrease the counter $c - 1$ times which is possible if and only if $c < i$. By the semantics of QATL this is exactly the case when $\mathcal{M}, s, i \models r < c$.

$\varphi = (r \leq c)$: The argument is similar to the case above.

$\varphi = (r \equiv_k c)$: In this case, Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(r \equiv_k c)$ if and only if he has a winning strategy where he subtracts one from the counter every time he can. The same is the case for Falsifier. For Verifier this is a winning strategy exactly when $\mathcal{M}, s, i \models (r \equiv_k c)$ if and only if $i \equiv_k c$. The reason is that after subtracting from the counter i times, the current state will be u_{k-1} if and only if

$k - 1 \equiv c - 1 - i \pmod{k}$

14

$\Leftrightarrow k \equiv c - i \pmod{k}$

$\Leftrightarrow i \equiv c \pmod{k} \Leftrightarrow i \equiv_k c$

Next, we consider the inductive cases.

$\varphi = \varphi_1 \vee \varphi_2$: Clearly, if Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$ or in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_2)$ then he has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1 \vee \varphi_2)$ since he can choose which of the games to play and reuse the winning strategy. On other hand, if Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1 \vee \varphi_2)$ then he is either winning in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$ or in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_2)$ because he can reuse the strategy and be sure to win in at least one of these games. Then, by using the induction hypothesis we have that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1 \vee \varphi_2)$ if and only if he has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$ or in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_2)$ if and only if $\mathcal{M}, s, i \models \varphi_1$ or $\mathcal{M}, s, i \models \varphi_2$ if and only if $\mathcal{M}, s, i \models \varphi_1 \vee \varphi_2$.

$\varphi = \neg \varphi_1$: The construction essentially switches Verifier with Falsifier when creating $\mathcal{G}_{\mathcal{M},s,i}(\neg \varphi_1)$ from $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$. This means that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\neg \varphi_1)$ if and only if Falsifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$. As a consequence of the determinacy result for Borel games [20] we have that one-counter games with parity conditions are determined. It follows that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\neg \varphi_1)$ if and only if Verifier does not have a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\varphi_1)$. Using the induction hypothesis this means that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\neg \varphi_1)$ if and only if $\mathcal{M}, s, i \not\models \varphi_1$ if and only if $\mathcal{M}, s, i \models \neg \varphi_1$.

$\varphi = \langle\!\langle A \rangle\!\rangle \mathbf{X} \varphi_1$: There are two cases to consider. First, suppose $s \in S_j$ for some $j \in A$. Then Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \mathbf{X} \varphi_1)$ if and only if there is a transition $(s, v, s') \in R$ with $v + i \geq 0$ such that Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s',i+v}(\varphi_1)$ since parity objectives are prefix independent. Using the induction hypothesis, this is the case if and only if there is a transition $(s, v, s') \in R$ with $v + i \geq 0$ such that $\mathcal{M}, s', i + v \models \varphi_1$ which is the case if and only if $\mathcal{M}, s, i \models \langle\!\langle A \rangle\!\rangle \mathbf{X} \varphi_1$. For the case where $s \notin S_j$ for all $j \in A$ the proof is similar, but uses universal quantification over the transitions.

$\varphi = \langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi_1$: The intuition of the construction is that Verifier controls the players in A and Falsifier controls the players in $\Pi \setminus A$. At each configuration $(s', v) \in S \times \mathbb{N}$ of the game Falsifier can challenge the truth value of φ_1 by going to $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ in which Falsifier has a winning strategy if and only if φ_1 is indeed false in $\mathcal{M}, s', v$. If Falsifier challenges at the wrong time or never challenges then Verifier can make sure to win.

More precisely, suppose Verifier has a winning strategy σ in $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi_1)$ then every possible play when Verifier plays according to σ either never goes into one of the modules $\mathcal{G}_{\mathcal{M},s'}(\varphi_1)$ or the play goes into one of the modules at some point and never returns. Since σ is a winning strategy for I, we have by the induction hypothesis that every pair $(s', v) \in S \times \mathbb{N}$ reachable when Verifier plays according to σ is such that $\mathcal{M}, s', v \models \varphi_1$, because otherwise σ would not be a winning strategy for I. If coalition A follows the same strategy σ adapted to $\mathcal{M}$ then the same state, value pairs are reachable. Since for all these reachable pairs (s', v) we have $\mathcal{M}, s', v \models \varphi_1$ this strategy is a witness

that $\mathcal{M}, s, i \models \langle\!\langle A \rangle\!\rangle \mathbf{G} \varphi_1$.

On the other hand, suppose that coalition A can ensure $\mathbf{G}\varphi_1$ from (s, i) using strategy σ. Then in every reachable configuration (s', v) we have $\mathcal{M}, s', v \models \varphi_1$. From this we can generate a winning strategy for Verifier in $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1)$ that plays in the same way until (if ever) Falsifier challenges and takes a transition to a module $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ for some (s', v). Since the same configurations can be reached before a challenge as when A plays according to σ, this means that Verifier can make sure to win in $\mathcal{G}_{\mathcal{M},s',v}(\varphi_1)$ by the induction hypothesis. Thus, if Falsifier challenges Verifier can make sure to win and if Falsifier never challenges Verifier also wins since all states reached have color 0. Thus, Verifier has a winning strategy in $\mathcal{G}_{\mathcal{M},s,i}(\langle\!\langle A \rangle\!\rangle \mathbf{G}\varphi_1)$.

$\varphi = \langle\!\langle A \rangle\!\rangle \varphi_1 \mathbf{U} \varphi_2$: The proof works as the case above with some minor differences. In this case, Verifier needs to show that he can reach a configuration where φ_2 is true when controlling the players in A and therefore he loses if he can never reach a module $\mathcal{G}_{\mathcal{M},s',v}(\varphi_2)$ such that $\mathcal{M}, s', v \models \varphi_2$. At the same time, he has to make sure that configurations (s', v) where $\mathcal{M}, s', v \not\models \varphi_1$ are not reached in an intermediate configuration since Falsifier still has the ability to challenge, as in the previous case. Note that Verifier gets the chance to commit to showing that φ_2 is true in a given configuration before Falsifier gets the change to challenge the value of φ_1. This is due to the definition of the until operator that does not require φ_1 to be true at the point where φ_2 becomes true. We leave out the remaining details.

$\square$

B Full proof of Theorem 7

We will show that model-checking ATL* in OCGMs is 2ExpSpace-hard by a reduction from the word acceptance problem for a deterministic doubly-exponential space Turing machine. From this, the theorem follows from the observations in the main text.

Let $\mathcal{T} = (Q, q_0, \Sigma, \delta, q_F)$ be a deterministic Turing machine that uses at most $2^{2^{|w|^k}}$ tape cells on input w where k is a constant and $|w|$ is the number of symbols in w. Here, Q is a finite set of control states, $q_0 \in Q$ is the initial control state. $\Sigma = \{0, 1, \#, a, r\}$ is the tape alphabet containing the blank symbol $\#$ and special symbols a and r such that $\mathcal{T}$ accepts immediately if it reads a and rejects immediately if it reads r, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{\text{Left}, \text{Right}\}$ is the transition function and $q_F \in Q$ is the accepting state. If $\delta(q, a) = (q', a', x)$ we write $\delta_1(q, a) = q', \delta_2(q, a) = a'$ and $\delta_3(q, a) = x$. Let $\Sigma_I = \Sigma \setminus \{\#\}$. Now, let $w = w_1...w_{|w|} \in \Sigma_I^*$ be an input word. From this we construct an OCGM $\mathcal{M}$, an initial state s_0 and a QATL* formula Φ all with size polynomial in $n = |w|^k$ and $|\mathcal{T}|$ such that $\mathcal{T}$ accepts w if and only if $\mathcal{M}, (s_0, 0) \models \Phi$.

We use an intermediate step in the reduction for simplicity of the arguments. This is done by considering an OCG $\mathcal{G} = (S', \{\text{Verifier}, \text{Falsifier}\}, (S'_{\text{Verifier}}, S'_{\text{Falsifier}}), R')$ with two players Verifier and Falsifier and an initial state s'_0 such that Verifier can force the play to reach s'_F if and only if $\mathcal{T}$ accepts w. However, the size of the set S' of states will be doubly-exponential in n. The idea of this construction resembles a reduction from the word acceptance problem for polynomial-space Turing machines to the emptiness problem for alternating finite automata with

a singleton alphabet used in [16]. Afterwards we will reduce this to model-checking of the ATL* formula Φ in $\mathcal{M}$ where $|S|$ is polynomial in n. This reduction can be performed by considering a more involved formula. We will use a technique similar to those used in [17] and [5] to simulate a 2^n-bit counter by using LTL properties and alternation between the players. This is the main trick to keep the state-space of $\mathcal{M}$ small.

We start with some notation. We assume that $\mathcal{T}$ uses the tape cells numbered $1, ..., 2^{2^n}$ and that the tape head points to position 1 initially. In addition, suppose for ease of arguments that there are two extra tape cells numbered 0 and $2^{2^n} + 1$ such that $\mathcal{T}$ immediately accepts if the tape head reaches cell 0 or cell $2^{2^n} + 1$. That is, cell 0 and $2^{2^n} + 1$ holds the symbol a initially. Further, assume without loss of generality that if $\mathcal{T}$ halts it always does so with the tape head pointing to cell 1 that contains the symbol a. Since $\mathcal{T}$ is deterministic it has a unique (finite or infinite) run on the word w which is a sequence $C_0^w C_1^w ...$ of configurations. Let $\Delta = \Sigma \cup (Q \times \Sigma)$. Then each configuration C_i^w is a sequence in $\Delta^{2^{2^n}+2}$ containing exactly one element in $Q \times \Sigma$ which is used to specify the current control state and location of the tape head. For instance, the initial configuration C_0^w is given by

$$C_0^w = a(q_0, w_1)w_2 w_3 ... w_{|w|} \#\#....\#a$$

We use $C_i^w(j)$ to denote the jth element of configuration C_i^w. For a given element $d \in \Delta$ we define the set $\mathrm{Pre}(d)$ of predecessor triples of d as

$$
\begin{aligned}
\mathrm{Pre}(d) = \quad & \{(d_1, d_2, d_3) \in \Sigma^3 \mid d_2 = d\} \\
& \cup \{((q, b), d_2, d_3) \in (Q \times \Sigma) \times \Sigma^2 \mid d = (\delta_1(q, b), d_2) \text{ and } \delta_3(q, b) = \mathrm{Right}\} \\
& \cup \{((q, b), d_2, d_3) \in (Q \times \Sigma) \times \Sigma^2 \mid d = d_2 \text{ and } \delta_3(q, b) \neq \mathrm{Right}\} \\
& \cup \{(d_1, d_2, (q, b)) \in \Sigma^2 \times (Q \times \Sigma) \mid d = (\delta_1(q, b), d_2) \text{ and } \delta_3(q, b) = \mathrm{Left}\} \\
& \cup \{(d_1, d_2, (q, b)) \in \Sigma^2 \times (Q \times \Sigma) \mid d = d_2 \text{ and } \delta_3(q, b) \neq \mathrm{Left}\} \\
& \cup \{(d_1, (q, b), d_3) \in \Sigma \times (Q \times \Sigma) \times \Sigma \mid d = \delta_2(q, b)\}
\end{aligned}
$$

The idea is that given the three elements $C_i^w(j-1), C_i^w(j)$ and $C_i^w(j+1)$ one can uniquely determine $C_{i+1}^w(j)$ according to the definition of a Turing machine. $\mathrm{Pre}(d)$ is then the set of all triples (d_1, d_2, d_3) such that it is possible to have $C_i^w(j-1) = d_1, C_i^w(j) = d_2, C_i^w(j+1) = d_3$ and $C_{i+1}^w(j) = d$.

We now define the OCG $\mathcal{G} = ((S', \{\mathrm{Verifier}, \mathrm{Falsifier}\}, (S'_{\mathrm{Verifier}}, S'_{\mathrm{Falsifier}}), R'))$ by

- $S' = (\{0, ..., 2^{2^n} + 1\} \times (\Delta \cup \Delta^3)) \cup \{s_0', s_z', s_r', s_F'\}$

- $S'_{\mathrm{Verifier}} = (\{0, ..., 2^{2^n} + 1\} \times \Delta) \cup \{s_0'\}$

- $S'_{\mathrm{Falsifier}} = (\{0, ..., 2^{2^n} + 1\} \times \Delta^3) \cup \{s_z', s_r', s_F'\}$

- R' is the least relation such that

    - $(s_0', 1, s_0') \in R'$
    - $(s_0', 0, (1, (q_F, a))) \in R'$
    - $((j, d), 0, (j, (d_1, d_2, d_3))) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $(d_1, d_2, d_3) \in \mathrm{Pre}(d)$
    - For $j \in \{0, 2^{2^n} + 1\}$ we have $((j, a), 0, s_F') \in R'$ and $((j, d), 0, s_r') \in R'$ when $d \neq a$
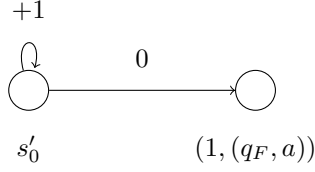
17

Figure 10: From the initial state, Verifier can increase the counter to any natural number before starting the game.

- $((j, d), 0, s'_z) \in R'$ for all (j, d) such that $C_0^w(j) = d$.
- $(s'_z, 0, s'_F) \in R'$
- $(s'_z, -1, s'_r) \in R'$
- $((j, (d_1, d_2, d_3)), -1, (j - 1, d_1)) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $d_1, d_2, d_3 \in \Delta$
- $((j, (d_1, d_2, d_3)), -1, (j, d_2)) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $d_1, d_2, d_3 \in \Delta$
- $((j, (d_1, d_2, d_3)), -1, (j + 1, d_3)) \in R'$ for all $j \in \{1, ..., 2^{2^n}\}$ and all $d_1, d_2, d_3 \in \Delta$

The different types of transitions are shown in Figure 10, 11 and 12. The intuition is that Verifier tries to show that $\mathcal{T}$ accepts w and Falsifier tries to prevent this. Initially, Verifier can increase the counter to any natural number, assume he chooses v. If $\mathcal{T}$ accepts w it does so in a final configuration with the tape head pointing at cell 1 holding the symbol a with the current control state q_F. The game is now played by moving backwards from the state $(1, (q_F, a))$ holding this information. Verifier can choose a predecessor triple that leads to $(1, (q_F, a))$. Player Falsifier then chooses one of the elements of the triple, the counter is decreased by one and the play continues like this. Finally, if the counter is 0 in a state (j, d) such that $C_0^w(j) = d$ then Verifier can win by going to s'_z from which Falsifier can only go to s'_F. We will argue that Verifier can make sure that this happens if and only if $\mathcal{T}$ accepts w after performing v steps.

Lemma 10. *The configuration $((j, d), i) \in (\{1, ..., 2^{2^n}\} \times \Delta) \times \mathbb{N}$ is winning for Verifier if and only if $C_i^w(j) = d$. In particular $((1, (q_F, a)), i)$ is winning for Verifier if and only if $C_i^w(1) = (q_F, a)$ if and only if $\mathcal{T}$ accepts w after i steps of computation.*

Proof. The proof is done by induction on i. For the base case $i = 0$ the statement says that $((j, d), 0)$ is winning for Verifier if and only if $C_0^w(j) = d$. Indeed, if $((j, d), 0)$ is winning for Verifier he must go directly from (j, d) to s'_z because all other paths are blocked after one step since the counter value is 0. If he goes to s'_z then he wins because Falsifier can only go to s'_F. However, note that there is only a transition from (j, d) to s'_z if $C_0^w(j) = d$ by construction. Thus, if Verifier is winning from $((j, d), 0)$ then $C_0^w(j) = d$. For the other direction, suppose $C_0^w(j) = d$. Then Verifier can make sure to win by going to s'_z.

For the induction step, suppose the lemma is true for i. Now we need to show that $((j, d), i+1)$ is winning for Verifier if and only if $C_{i+1}^w(j) = d$. Suppose
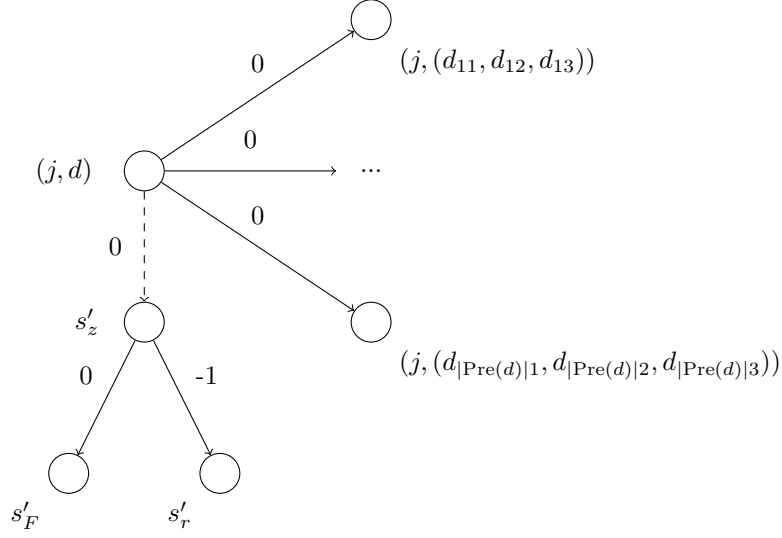
18

Figure 11: From a state $(j, d) \in \{1, ..., 2^{2^n}\} \times \Delta$ Verifier can choose a predecessor triple of d. The dashed transition is enabled only when $C_0^w(j) = d$. In this case Verifier can be sure to win if the current counter value is 0.
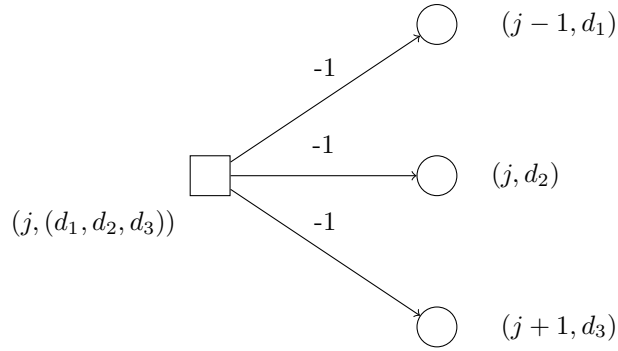

Figure 12: From a precedessor triple chosen by Verifier, Falsifier can choose which predecessor to continue with.

first that $((j, d), i + 1)$ is winning for Verifier. The winning strategy σ cannot consist in going directly to s'_z because then Falsifier can go to s'_r. Thus, Verifier must choose a predecessor triple $(d_1, d_2, d_3) \in \text{Pre}(d)$ when playing according to σ. After he chooses this, Falsifier chooses one of them and the counter is decreased by one. Thus, Falsifier can choose either $((j - 1, d_1), i)$, $((j, d_2), i)$ or $(j+1, d_3), i)$. Thus, by the induction hypothesis $C_i^w(j-1) = d_1$, $C_i^w(j) = d_2$ and $C_i^w(j+1) = d_3$ since Verifier is winning. By the definition of predecessor triples, this means that $C_{i+1}^w(j) = d$. For the other direction, suppose $C_{i+1}^w(j) = d$. Then by going to the state $(j, (C_i^w(j - 1), C_i^w(j), C_i^w(j + 1)))$ he can be sure to win by the induction hypothesis. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Lemma 11. *Starting in configuration $(s'_0, 0)$ Verifier can make sure to reach s'_F if and only if $\mathcal{T}$ accepts w.*

We have now reduced the word acceptance problem to a reachability game in an OCG $\mathcal{G}$ with a doubly-exponential number of states. Due to the structure of $\mathcal{G}$ we can reduce this to model-checking the ATL* formula Φ in the OCGM $\mathcal{M}$. The difficult part is that we need to store the number of the tape cell that the tape head is pointing at, which can be of doubly-exponential size. The other features of $\mathcal{G}$ are polynomial in the input. Note that at each step of the game, the position of the tape head either stays the same, increases by one or decreases by one. This is essential for our ability to encode it using ATL*. We construct $\mathcal{M}$ much like $\mathcal{G}$ but where the position of the tape head is not present in the set of states. Instead, for each transition in the game between states s and s' we have a module in which Verifier encodes the position of the tape head by his choices. At the same time, Falsifier has the possibility to challenge if Verifier has not chosen the correct value of the tape head position. This can be ensured by use of the ATL* formula $\Phi = \langle\!\langle \{\text{Verifier}\} \rangle\!\rangle \varphi$ where φ is an LTL formula. The details of simulating a 2^n-bit counter like this can be obtained from [17, 5]. According to the choices of Falsifier then Verifier must be able to increase, decrease or leave unchanged the position of the tape head. This can be enforced by a formula with a size polynomial in n. Except for having to implement the position of the tape head in this way, the rules of $\mathcal{M}$ are the same as for $\mathcal{G}$ where Verifier needs to show that $\mathcal{T}$ accepts w by choosing a strategy that ensures reaching a certain state in the game while updating the tape head position correctly. In the end, this means that for the initial state s_0 in $\mathcal{M}$ corresponding to s'_0 in $\mathcal{G}$ we get $\mathcal{M}, s_0, 0 \models \langle\!\langle \{\text{Verifier}\} \rangle\!\rangle (\varphi \wedge \mathbf{F} s_F)$ if and only if $\mathcal{T}$ halts on w. Here we assume that the play also goes to a halting state s_F corresponding to s'_F if Falsifier challenges the counter value incorrectly.

Theorem 7. *The combined complexity of model-checking* QATL* *is* 2ExpSpace*-complete for both OCGMs and SOCGMs. The data complexity of model-checking* QATL* *is* PSpace*-complete for OCGMs and* ExpSpace*-complete for SOCGMs.*

References

[1] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[2] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin. Synthesis from ltl specifications with mean-payoff objectives. In *TACAS*, pages 169–184, 2013.

[3] Patricia Bouyer, Ulrich Fahrenberg, Kim Guldstrand Larsen, Nicolas Markey, and Jirí Srba. Infinite runs in weighted timed automata with energy constraints. In *FORMATS*, pages 33–47, 2008.

[4] Laura Bozzelli. Complexity results on branching-time pushdown model checking. *Theor. Comput. Sci.*, 379(1-2):286–297, 2007.

[5] Laura Bozzelli, Aniello Murano, and Adriano Peron. Pushdown module checking. In *LPAR*, pages 504–518, 2005.

[6] Tomás Brázdil, Petr Jancar, and Antonín Kucera. Reachability games on extended vector addition systems with states. In *ICALP (2)*, pages 478–489, 2010.

[7] Nils Bulling and Valentin Goranko. How to be both rich and happy: Combining quantitative and qualitative strategic reasoning about multi-player games (extended abstract). In *SR*, pages 33–41, 2013.

[8] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theor. Comput. Sci.*, 458:49–60, 2012.

[9] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.

[10] Stéphane Demri and Régis Gascon. The effects of bounding syntactic resources on presburger ltl. *J. Log. Comput.*, 19(6):1541–1575, 2009.

[11] E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.

[12] Javier Esparza, Antonín Kucera, and Stefan Schwoon. Model-checking ltl with regular valuations for pushdown systems. In *TACS*, pages 316–339, 2001.

[13] Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. *Electr. Notes Theor. Comput. Sci.*, 9:27–37, 1997.

[14] Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Model checking succinct and parametric one-counter automata. In *ICALP (2)*, pages 575–586, 2010.

[15] Stefan Göller and Markus Lohrey. Branching-time model checking of one-counter processes and timed automata. *SIAM J. Comput.*, 42(3):884–923, 2013.

[16] Petr Jancar and Zdenek Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Inf. Process. Lett.*, 104(5):164–167, 2007.

[17] Orna Kupferman, P. Madhusudan, P. S. Thiagarajan, and Moshe Y. Vardi. Open systems in reactive environments: Control and synthesis. In *CONCUR*, pages 92–107, 2000.

[18] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. Module checking. *Inf. Comput.*, 164(2):322–344, 2001.

[19] Christof Löding, P. Madhusudan, and Olivier Serre. Visibly pushdown games. In *FSTTCS*, pages 408–420, 2004.

[20] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, September 1975.

[21] Nir Piterman. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.

[22] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[23] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190, 1989.

[24] Amir Pnueli and Roni Rosner. On the synthesis of an asynchronous reactive module. In *ICALP*, pages 652–671, 1989.

[25] Olivier Serre. Parity games played on transition graphs of one-counter processes. In *FoSSaCS*, pages 337–351, 2006.

[26] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *ICALP*, pages 628–641, 1998.

[27] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.

[28] Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *FOCS*, pages 185–194, 1983.