Solving Counter Parity Games

Dietmar Berwanger[1], Łukasz Kaiser[2], and Simon Leßenich[1,3]⋆

[1] LSV, CNRS & ENS Cachan, France
[2] LIAFA, CNRS & Université Paris Diderot – Paris 7, France
[3] Mathematische Grundlagen der Informatik, RWTH Aachen, Germany

Abstract. We study a class of parity games equipped with counters that evolve according to arbitrary non-negative affine functions. These games capture several cost models for dynamic systems from the literature. We present an elementary algorithm for computing the exact value of a counter parity game, which both generalizes previous results and improves their complexity. To this end, we introduce a class of ω-regular games with imperfect information and imperfect recall, solve them using automata-based techniques, and prove a correspondence between finite-memory strategies in such games and strategies in counter parity games.

1 Introduction

Games with ω-regular winning conditions, and especially parity games, are a fundamental model for program verification and synthesis [12]. Such winning conditions allow to express reachability, safety, and liveness properties. However, when specifying, for instance, that for each request Q_i there will be finally a response R_i, one is often interested not only in the existence of a response R_i – a *qualitative* property, but also that the response will occur in at most k seconds after the request – a *quantitative* constraint.

Quantitative questions about reactive systems have been approached in several ways. One possibility is to extend a temporal logic with new, quantitative operators as, for instance, the "prompt" operator for LTL proposed in [11]. While the existence of a response R_i is formulated in LTL by $\mathsf{F}R_i$, the Prompt-LTL formula $\mathsf{F_p}R_i$ expresses that the waiting time is bounded. Realizability for this logic was solved in [11] and optimal bounds on the waiting time for Prompt-LTL formulas were established in [14]. Another possibility is to consider formulas which evaluate to numbers rather than truth values. The quantitative version of CTL with discounts studied in [6], and the quantitative μ-calculus investigated in [7] follow this direction.

Both model-checking and realizability problems for most of these logics are reduced to solving games with additional quantitative features. Several classes of such games have therefore been investigated [3,4,5,9], to provide better algorithms for existing logics and to suggest new formalisms with good algorithmic properties. One relevant example is the synthesis of optimal strategies in

⋆ This author was supported by the ESF Research Networking Programme GAMES.

request-response games [9]. Here, the problem of minimizing the waiting time for a response is investigated, considering different ways to accumulate waiting costs. In one model, the penalty for waiting k many steps is k^2, to discourage long waiting times. This illustrates that cost functions that depend only linearly on the time may not be sufficient for certain applications.

In this work, we introduce a model of counter parity games in which counters are updated by arbitrary non-negative affine transformations along the moves of a play. When a play ends, the counters are used to determine the payoff, whereas on infinite plays, a parity condition is applied. For example, the time since the last request Q_i can be stored in the counter c_i, which will be reset every time the response R_i arrives. The maximum value reached by c_i is then the maximal waiting time for R_i. Similarly, one can express arbitrary Prompt-LTL conditions. It is also possible to have another counter, d_i, which will increase *by* c_i every time the request Q_i is active, and will also be reset on R_i. Note that this is an affine update and that $c_i + 2d_i$ stores the waiting time squared, and thus allows to simulate the cost model from [9]. Affine functions also allow to swap counters and multiply them by constants, which can be used, e.g., to model process migration and pricing. This could be used to extend the scope of formal analysis of online algorithms [1]. Moreover, counter parity games are a strict generalization of counter-reset games used in [8] to approximate the quantitative μ-calculus over a class of hybrid systems. For model-checking this logic, only a non-elementary algorithm was known so far [8] – our results both allow for a more general class of games and provide better complexity bounds. In [10], counter parity games are used for proving decidability of the counting μ-calculus, an extension of the quantitative μ-calculus to structured transition systems, such as graphs generated by regular tree grammars or by pushdown automata.

2 Counter Parity Games

To define counter parity games, let us fix a natural number k of counters and let $\mathcal{F}_k$ be the set of k-dimensional affine functions f over non-negative integers:

$$f : \mathbb{N}^k \to \mathbb{N}^k, \quad f(c) = A \cdot c + B$$

for some matrix $A \in \mathbb{N}^{k \times k}$ and some vector $B \in \mathbb{N}^k$. Note that we only consider functions $\mathbb{N}^k \to \mathbb{N}^k$, thus the coefficients are also assumed to be non-negative.

A *counter parity game* $\mathcal{G} = (V, V_{\max}, V_{\min}, E, \Omega, \lambda)$ with k counters is played by two players, Maximizer and Minimizer, on a directed graph (V, E). The vertex set is partitioned into vertices $V_{\max}$ of Maximizer and vertices $V_{\min}$ of Minimizer. Vertices are colored by the priority function $\Omega : V \to \{0, \ldots, d-1\}$, edges are labeled by affine functions, i.e., $E \subseteq V \times \mathcal{F}_k \times V$, and terminal vertices $T = \{v \mid vE = \emptyset\}$ are labeled by $\lambda : T \to \{+, -\} \times \{0, \ldots, k-1\}$.

The k counters are represented by a vector $c \in \mathbb{N}^k$ of k natural numbers. We write c_i for the i-th component of c, i.e., the i-th counter. At the beginning of a play, all counters are 0, thus $c = 0^k$. Throughout a play, counters are updated according to the labels of the edges: if the current value of the counter vector is c and an edge (u, f, v) is taken, then the new value is $f(c)$. Maximizer moves at positions $V_{\max}$, while Minimizer moves at $V_{\min}$.

A *play* $\pi = v_0 f_0 v_1 f_1 v_2 \ldots$ is a sequence of vertices and edge labels such that, for each $i \geq 0$, $(v_i, f_i, v_{i+1}) \in E$. For infinite plays π, the payoff $p(\pi)$ is determined by the parity condition given by Ω: it is $-\infty$ if the minimal priority seen infinitely often in $\Omega(v_0)\Omega(v_1)\ldots$ is odd, and ∞ if it is even. Finite plays π end at a terminal vertex t and $p(\pi)$ is determined by $\lambda(t)$ and the current counters: it is $s\,c_i$ if $\lambda(t) = (s, i)$ and the current vector is c. The objective of Maximizer is to maximize the payoff, whereas Minimizer seeks to minimize it.

A *strategy* of Maximizer is a function $f : (V\mathcal{F}_k)^* V_{\max} \to \mathcal{F}_k \times V$, such that, for each prefix π of a play, if $f(\pi v) = (f, w)$ then $(v, f, w) \in E$; analogously, we define strategies of Minimizer $f : (V\mathcal{F}_k)^* V_{\min} \to \mathcal{F}_k \times V$. We say that f uses *memory* M if there exists a $m_0 \in M$, a function $\mathrm{update} : M \times \mathcal{F}_k \times V \to M$, and a function $f_M : M \times V \to \mathcal{F}_k \times V$ such that $f(v_0 f_0 v_1 \ldots f_{n-1} v_n) = f_M(\mathrm{update}^*(v_0 f_0 v_1 \ldots f_{n-1} v_n, m_0), v_n)$, where update^* is defined inductively by $\mathrm{update}^*(\varepsilon, m) = m$ and

$$\mathrm{update}^*(v_0 f_0 v_1 \ldots f_k v_{k+1}, m) = \mathrm{update}(\mathrm{update}^*(v_0 f_0 v_1 \ldots v_k, m), f_k, v_{k+1}).$$

The *size* of the memory is $|M|$ and a *finite-memory strategy* is one that uses a finite memory M.

Strategies can be identified with labelings of the infinite tree $\mathcal{T}(\mathcal{G}, v_0)$ obtained by unfolding the arena of $\mathcal{G}$ from v_0. This allows to speak about *regular* sets of strategies, i.e., sets which are recognized by non-deterministic parity tree automata over $\mathcal{T}(\mathcal{G}, v_0)$. We assume that the reader is familiar with this standard way of identifying strategies with labelings of the infinite tree. We will also use algorithms for alternating automata on infinite trees, which are more precisely recalled in the technical report [2].

A counter parity game $\mathcal{G}$ is *determined* if the supremum of the payoffs that Maximizer can achieve coincides with the infimum of the payoffs that the Minimizer cannot avoid, that is, if

$$\sup_{f \in \Sigma_{\max}} \inf_{g \in \Sigma_{\min}} p(\alpha_{f,g}(v)) = \inf_{g \in \Sigma_{\min}} \sup_{f \in \Sigma_{\max}} p(\alpha_{f,g}(v)) =: \mathrm{val}\,\mathcal{G}(v),$$

where $\Sigma_{\max}$ and $\Sigma_{\min}$ are the sets of all strategies of Maximizer and Minimizer, and $\alpha_{f,g}(v)$ is the unique play consistent with both f and g.

As counter parity games are a special case of quantitative parity games on infinite arenas (we can encode counter values in the vertices and adjust the edges accordingly), and it was shown in [7] that quantitative parity games are determined on arenas of arbitrary size, we obtain the following corollary.

Proposition 1 ([7]). *Every counter parity game $\mathcal{G}$ is determined: for each vertex v the value* $\mathrm{val}\,\mathcal{G}(v)$ *exists.*

However, these results do not imply that the value of a counter parity game can actually be computed, nor do they give any insight into the structure of strategies in the game. Our main technical result, stated below, identifies good regular over-approximations for the strategies of Minimizer. Let us denote by $2\mathrm{EXP}(f)$ the family of functions $2^{2^{\mathcal{O}(f)}}$ and, for a counter parity game $\mathcal{G}$ and a strategy g of Minimizer, let us write $\mathrm{val}_g\,\mathcal{G}(v) = \sup_{f \in \Sigma_{\max}} p(\alpha_{f,g}(v))$ for the supremum of all payoffs Maximizer can get when playing against g.

Theorem 2. *Let $\mathcal{G} = (V, V_{\max}, V_{\min}, E, \Omega, \lambda)$ be a counter parity game with k counters, and let $v_0 \in V$. One can compute a constant $m = 2\mathrm{EXP}(k)$ and a regular set Σ of strategies of Minimizer, recognized by a non-deterministic parity tree automaton of size $2\mathrm{EXP}((|V|+km)^3)$ and with polynomial index, such that:*

– *for every strategy $g \notin \Sigma$, $\mathrm{val}_g\,\mathcal{G}(v_0) = \infty$, and*
– *for every memory M strategy $g \in \Sigma$, $\mathrm{val}_g\,\mathcal{G}(v_0) < 2\mathrm{EXP}(|M|^2 \cdot (|V| \cdot 2^k)^4)$.*

Note that the set Σ is only an over-approximation of the strategies of the Minimizer which guarantee a bounded payoff – only finite-memory strategies from Σ have this property. The following example illustrates why this is a crucial constraint. It also shows that the exact set of strategies which guarantee a bounded payoff is not regular, which is why we compute an over-approximation.

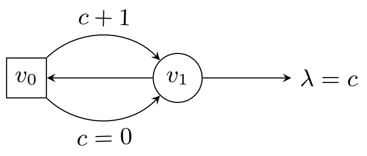


Fig. 1. Counter Parity Game with Value 0.

Example 3. Let $\mathcal{G}$ be the 1-counter parity game depicted in Figure 1. Minimizer's vertices are drawn as squares and Maximizer's as circles. The game proceeds as follows: Minimizer can either increment the single counter c or reset it, and then Maximizer can decide to continue or to exit and take the current value of c. The priorities in the game are all odd, thus Maximizer must exit at some point.

Clearly, Minimizer can reset c at each step and thus $\mathrm{val}\,\mathcal{G}(v_0) = 0$. But which strategies g of Minimizer guarantee $\mathrm{val}_g\,\mathcal{G}(v_0) < \infty$? Of course, these are exactly the strategies that do not allow c to grow beyond some bound B, i.e., alternate the two possible moves $(c = 0)$ and $(c + 1)$ according to the pattern

$$(c+1)^{n_1}(c=0)^+(c+1)^{n_2}(c=0)^+(c+1)^{n_3}(c=0)^+\cdots,$$

such that, for some $B \in \mathbb{N}$, $n_i < B$ for all i. But this set, i.e., the set of all strategies g which guarantee that $\mathrm{val}_g\,\mathcal{G}(v_0) < \infty$, is not regular.

To over-approximate the strategies which guarantee a bounded payoff, we will put in Σ all strategies that take the reset move $(c = 0)$ infinitely often.

Note that this disregards the restriction on the number of $(c+1)$-moves between resets. All *finite-memory* strategies from Σ indeed guarantee a bounded value, because the number of increments between resets cannot exceed the memory size. However, consider the infinite memory strategy g_* which plays according to the pattern above with $n_i = i$. This strategy is in Σ, but $\operatorname{val}_{g_*} \mathcal{G}(v_0) = \infty$.

We prove Theorem 2 in the following sections. Let us first state that it can be used to decide boundedness and compute the value of counter parity games.

Corollary 4. *Given a finite counter parity game $\mathcal{G}$ with initial vertex v, one can decide whether* $\operatorname{val} \mathcal{G}(v) = \infty$ *in* 2EXPTIME*, if the number of counters is fixed, and in* 4EXPTIME *otherwise. The value* $\operatorname{val} \mathcal{G}(v)$ *can be computed exactly in* 4EXPTIME *if the number of counters is fixed and in* 6EXPTIME *otherwise.*

3 Marks for Counter Updates

In this section, we make the first step towards proving Theorem 2 and introduce *marks* for counter update functions. Marks are an abstraction and allow to determine whether a counter increased with respect to other counters or not.

Notation. When referring to a sequence s, we write $s[i]$ for the i-th element of s. We always count from 0, i.e., $s[0]$ is the first element of s. For a set $I \subseteq \mathbb{N}$ of indices, we write $s|^I$ to denote the sub-sequence of s consisting only of the elements with indices in I. We refer to a sequence of fixed finite length as a vector. For a vector s, we write $s^{>0}$ to denote the vector t with $t[i] := 1$ if $s[i] > 0$ and $t[i] := 0$ otherwise. Finally, we write $[n]$ to denote the set $\{0, \ldots, n-1\}$.

Let k be the dimension of the counter vector $c \in \mathbb{N}^k$. We consider all counter update functions $f : \mathbb{N}^k \to \mathbb{N}^k$ that admit marking in the following sense.

A *mark* is a mapping $m : \{0,1\}^k \times [k] \to \{\bot\} \uplus [k] \uplus \mathcal{P}([k])$. A function $f : \mathbb{N}^k \to \mathbb{N}^k$ *has mark* m if the following hold for all $c \in \mathbb{N}^k, i \in [k]$.

(i) If $m(c^{>0}, i) = \bot$ then $f(c)[i] = 0$.
(ii) If $m(c^{>0}, i) = j \in [k]$ then $f(c)[i] = c_j$.
(iii) If $m(c^{>0}, i) = D \in \mathcal{P}([k])$ and $D \neq \emptyset$ then $f(c)[i] > \max_{j \in D} c_j$.
(iv) If $m(c^{>0}, i) = \emptyset$ then $f(d)[i] = C > 0$ is constant for all d with $d^{>0} = c^{>0}$.
(v) $f(c)[i]$ depends only on the counters from $m(c^{>0}, i) = D$,
i.e., there exists a function f'_i such that $f(c)[i] = f'_i(c|^D)$.

Note that (iv) could be seen as special case of (v), but we distinguish whether the constant is 0, as in (i), or not. Intuitively, a mark determines, depending on which counters are 0 and which are not, whether the result will be 0, always stay equal to another counter, or increase over other counters.

In particular, if $m(d, i) = D$ then, after applying the counter update function, the (value of) counter i will be strictly greater than each of the counters from D. We write $m^{\geq}(d, i)$ for the set D of counters such that the value of c_i after the update will be greater or equal to the values of the counters in D, i.e.,

$m^{\geq}(d,i) := \emptyset$ if $m(d,i) = \bot$, $m^{\geq}(d,i) := \{l\}$ if $m(d,i) = l$, and $m^{\geq}(d,i) := D$ if $m(d,i) = D \in \mathcal{P}([k])$. Additionally, we write $m^{>0}(c^{>0})$ for the vector $d^{>0}$ if d results from the application of a function f with mark m to the vector c. Observe that $f(c)[i] = 0$ if, and only if, $m(c^{>0}, i) = \bot$ or $m(c^{>0}, i) = l$ and $c_l = 0$, and thus $m^{>0}(c^{>0}) = f(c)^{>0}$ is computable from $c^{>0}$ and m.

Example 5. Consider two counters c_0, c_1 and the update function f assigning $c_0 + c_1$ to c_0 and $2 \cdot c_0$ to c_1. This function has the following mark m: $m(0,0,i) = \bot$, $m(0,1,0) = 1$ as $c_0 + c_1 = c_1$ if $c_0 = 0$, and $m(1,0,0) = 0$ analogously; $m(0,1,1) = \bot$ as $2 \cdot 0 = 0$, but $m(1,0,1) = m(1,1,1) = \{0\}$ as $2 \cdot c_0 > c_0$ for $c_0 > 0$. Finally, $m(1,1,0) = \{0,1\}$ as c_0 exceeds both counters in this case.

Not only affine functions can be marked. For example, the function which updates c_i to $\max(c_j, c_l) + 1$ also has a mark, and thus our results also hold for such functions (in fact, a mark D expresses a lower bound of $\max D{+}1$). Note also that not all functions admit a marking. For example, if we updated c_1 to $c_0 \cdot c_1$ above, we would not be able to assign a mark. In particular, $m(1,1,i)$ is not definable, because whether the counter increases or stays unchanged depends on whether $c_i > 1$ and not just on whether $c_i > 0$. The methods we present generalize to more involved markings, but we do not introduce them here as we are interested in one class of functions, for which the above marks suffice.

Lemma 6. *Let $f : \mathbb{N}^k \to \mathbb{N}^k$ be affine. There exists a mark m_f for f.*

Another important property of marks (see [2] for the proofs) is that, when functions are composed, their marks can be composed as well.

Lemma 7. *Let f_1 and f_2 be counter update functions with marks m_1 and m_2. A mark $m = m_1 \circ m_2$ for $f(c) = f_2(f_1(c))$ can be computed from m_1 and m_2.*

Let us denote by $\mathcal{M}$ the set of all marks, which is finite, by definition. For a fixed number k of counters, $|\mathcal{M}| \leq \left(2^k + k + 1\right)^{2^{k + \log k}} = 2\mathrm{EXP}(k)$. Moreover, by the above lemma, the composition $\circ$ induces a computable finite semigroup structure on $\mathcal{M}$. It follows that languages of sequences of marks with definable properties are regular. For example, the language of all sequences $m_0 m_1 \ldots m_n \in \mathcal{M}^*$ such that $m = m_1 \circ \cdots \circ m_n$ satisfies, for a fixed C, i and d, that $C \subseteq m^{\geq}(d,i)$, is regular. This means that, for a fixed set of counters C and starting information about which counter is 0, we can determine in a regular way whether c_i will be at least as large as some counter from C.

To access marks, we extend counter parity games by the appropriate marking. Let $\mathcal{G}$ be a counter parity game with k counters. The *marked counter parity game* $\mathcal{G}_m = (V_m, V'_{\max}, V'_{\min}, E_m, \Omega_m, \lambda_m)$ is a game with k counters defined as follows.

- $V_m := V \times \{0,1\}^k$ (storing which counters are greater than 0).
- $V'_{\max} = \{(v,x) \in V_m \mid v \in V_{\max}\}$, $V'_{\min} = V_m \setminus V'_{\max}$.
- $E_m \subseteq V_m \times (\mathcal{F}_k \times \mathcal{M}) \times V_m$ stores the marks and updates the $c^{>0}$-vectors:
$$E_m := \{((u,x),(f,m_f),(v,m_f^{>0}(x))) \mid (u,f,v) \in E,\ x \in \{0,1\}^k\}.$$
- $\Omega_m(v,x) = \Omega(v)$ and $\lambda_m(v,x) = \lambda(v)$.

4 Second-Life Games

In this section, we introduce the class of second-life games with imperfect information and a specific kind of imperfect recall, which are essential for the construction in the next section. The construction of a second-life game starts with a game graph $\mathcal{G}$ with perfect information for two players, Player 0 and Player 1. The second-life game arena consists of several copies of this graph. Plays begin in the main instance of $\mathcal{G}$, which we call *first life*, and proceed as usual by moving a token along the edges of the graph. However, when Player 1 is in turn to move, he may switch to a copy of $\mathcal{G}$, a *second life*, without informing Player 0. If a terminal position is reached in the first life, the play simply ends. In contrast, if this happens in a second life, the play returns to the first life, and Player 0 forgets the part of the history spent in the second life. This part of the history is nevertheless relevant for the winning condition.

Let $\mathcal{G} = (V, V_0, V_1, E)$ be a game arena with $E \subseteq V \times A \times V$ for a set A of actions, and let $T = \{t \in V : tE = \emptyset\}$ denote the set of terminal vertices in $\mathcal{G}$. The *second-life game* $\mathcal{S}(\mathcal{G}, W)$ is a game with the set of actions

$$A_{\|} := A \cup \{\mathsf{Return}\} \cup \{\mathsf{Call}(a) \mid a \in A\}$$

and over the arena (V', V'_0, V'_1, E'), with

$$\begin{aligned}
V' &:= V \cup (V \times V);\\
V'_0 &:= V_0 \cup \{(u,v) \mid u \in V_0\} \cup \{(t,v) \mid t \in T, v \in V\}, \quad \text{and} \quad V'_1 := V' \setminus V'_0;\\
E' &:= E \cup \{((u,v), a, (u',v)) \mid (u,a,u') \in E\}\\
&\quad \cup \{(u, \mathsf{Call}(a), (v,v)) \mid u \in V_1, (u,a,v) \in E\} \qquad \text{(CALL)}\\
&\quad \cup \{((t,v), \mathsf{Return}, v) \mid t \in T\} \qquad \text{(RETURN)}
\end{aligned}$$

Winning conditions for second-life games have the form $W \subseteq A_{\|}^{\omega}$.

A play α is a – possibly infinite – alternating sequence of vertices and actions, $\alpha = v_0 a_0 v_1 a_1 v_2 \cdots$, such that $(v_i, a_i, v_{i+1}) \in E'$, for any index i. A finite play is one that ends at a terminal vertex. Every finite play is winning for Player 0; an infinite play is winning for Player 0 if, and only if, its action trace belongs to W.

Notice that all the moves of the arena $\mathcal{G}$ are available in the second-life game, regardless of whether the play is in the first or in a second-life copy. Additionally, when Player 1 moves at a vertex of the first-life copy $\mathcal{G}$, he can choose to switch to a second-life copy via a $\mathsf{Call}(\cdot)$ action.

The intended information structure of second-life games is captured by a constraint on strategies of Player 0. We postulate that Player 0 is not informed about whether the current vertex is in the main copy or in some other component. Furthermore, after any Call-Return sequence, Player 0 forgets the part of the play between Call and Return.

Here, and in the following, a Call-Return *sequence* is a sequence of the form

$$u \cdot \mathsf{Call}(a) \cdot (v,v) \cdot a_1 \cdot (v_1, v) \cdots (t,v) \cdot \mathsf{Return} \cdot v.$$

For any finite path π starting at a vertex v in the main copy, we define the path $\hat{\pi}$ obtained by replacing every Call-Return sequence $u \cdot \mathsf{Call}(a) \cdots \mathsf{Return} \cdot v$ by $u \cdot a \cdot v$, then replacing the remaining last $\mathsf{Call}(a)$ by a (if such a last Call exists), and finally projecting every occurring (u, v) to u.

Now, strategies of Player 0 are functions $f : (V' A_{\|})^* V'_0 \to A \times V$ such that, for every π ending in a vertex of Player 0 we have $f(\pi) = f(\hat{\pi})$. Thus, Player 0's strategies respect the information constraint described above. Strategies of Player 1 are not restricted in any way.

Notice that, for every play α, the sequence $\hat{\alpha}$ corresponds to a play in the main copy. However, W is given over $A_{\|}$. Nonetheless, Player 0 has no information about whether he is moving in one of the second-life components or in the main copy, and immediately after noticing that the play continues after a terminal (which means it must have been in a second-life component), he forgets this and all that happened in the component. Accordingly, any strategy of Player 0 can be viewed as a strategy over the vertex set V with actions A, i.e., as a strategy of Player 0 for the arena $\mathcal{G}$.

Our main result on second-life games, proved using automata techniques (c.f. [2]), states that the set of winning strategies of Player 0 is regular and an automaton recognizing it can be constructed effectively.

Theorem 8. *Let $\mathcal{G}$ be an arena with positions V and W a regular winning condition recognized by a deterministic parity automaton $\mathcal{A}$. The set of winning strategies of Player 0 in the second-life game $\mathcal{S}(\mathcal{G}, W)$ can be recognized by a non-deterministic parity tree automaton of size at most $2^{\mathcal{O}(|\mathcal{A}|^9 + |V|^3)}$.*

5 The Unboundedness Game

In the next step, we consider a marked counter parity game and check whether its value is unbounded, i.e., ∞, or not. To do this, we transform the marked game into a second-life game, where Minimizer takes the role of Player 0.

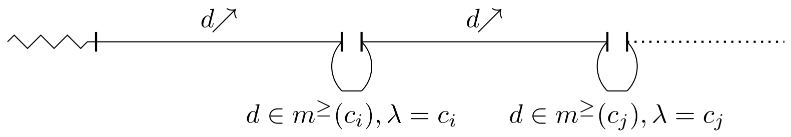
From the definition of the value of a counter parity game, there are two ways for the value to be ∞: Maximizer may have a winning strategy with respect to the parity condition, or a sequence $f_0, f_1, \cdots$ of strategies which ensure arbitrarily high payoffs. Via the reduction to second-life games, we combine the sequence of strategies for the latter situation into a single strategy. Intuitively, Maximizer will get the option to decide to try to reach a terminal position to "save" a payoff, and then continue increasing the counters. If in such a game Maximizer has a strategy to save higher and higher payoffs, or to win via the parity condition, this corresponds to a value of ∞. We exploit that marks form a finite semigroup to show that this can be formulated as a regular objective. Intuitively, the reason why we rely on second-life games with imperfect information and recall for Minimizer is that we need to avoid that Minimizer learns about whether Maximizer attempts to win by parity or by reaching arbitrarily high payoffs. If Minimizer had this information, he could adapt his strategy and neglect the other way of ensuring payoff ∞.

Let $\mathcal{G}_m$ be a marked counter parity game with arena $\mathcal{G}$ and terminal vertices T. The unboundedness game $\mathcal{G}^u$ is the second-life game $\mathcal{S}(\mathcal{G}_m, W)$ using $V_0 := V_{\min}$ and $V_1 := V_{\max}$ and with the winning condition W described below.

Recall that, if we remove all Call-Return sequences from a path in $\mathcal{G}^u$, we obtain a path in $\mathcal{G}_m$ that we call the main part. For better readability, we describe the winning condition in terms of both edge- and vertex-labels (functions/marks and priorities, respectively). Technically, this can be avoided by adding the color of the source vertex to the action label.

We describe the winning condition for Maximizer, i.e., Player 1, which is sufficient since regular languages are closed under complementation. Maximizer wins a play α if, and only if, the main copy is visited infinitely often, no terminal vertex inside the main copy is seen, and

– the main part satisfies the parity condition of $\mathcal{G}_m$, or
– there exists a counter d such that, from some point onwards, counter d is increased in the main part, then a Call is taken and a Return from a terminal where a payoff greater than d would be obtained in the original counter game, and after the Return this is repeated, ad infinitum.


$d\nearrow$
$d\nearrow$
$d \in m^{\geq}(c_i), \lambda = c_i$
$d \in m^{\geq}(c_j), \lambda = c_j$

By properties of marks, finite sequences of marks after which a counter d has been increased form a regular language $d\nearrow$. Also, finite sequences starting with a Call and ending with a Return from a vertex with $\lambda = c$ such that, for the sequence of marks in between, counter c is, at the end, greater than d at the beginning, form a regular language $c^{>d}$. Thus, the later part of W is the union of the main part satisfying the parity condition and the play being of the form $A_{\parallel}^* \cdot (d\nearrow \cdot c^{>d})^\omega$. This is ω-regular.

Let us calculate the size of the automaton for the above condition. To check the language $d\nearrow$, $|\mathcal{M}|$ states suffice for a deterministic finite word automaton (using composition on the marks), and the same holds for $c^{>d}$. Checking $d\nearrow \cdot c^{>d}$ can thus be done with $\mathcal{O}(|\mathcal{M}|)$ many states by a non-deterministic automaton. By considering Büchi acceptance with the same accepting states, we get a Büchi automaton for the language $(d\nearrow \cdot c^{>d})^\omega$ with $\mathcal{O}(|\mathcal{M}|)$ states. If we add a new initial state and take a copy of the automaton for $(d\nearrow \cdot c^{>d})^\omega$ for every $d < k$, we build a nondeterministic Büchi automaton of size $\mathcal{O}(k \cdot |\mathcal{M}|)$ which accepts a play if it is won via some counter. (It waits in the initial state until the actual d is correctly guessed and then moves to the respective copy.) For the parity part, we need an automaton of size $|\Omega(V)| < |V|$. Taking the union of the two, we get a non-deterministic parity automaton of size $\mathcal{O}(|V| + k|M|)$ and index $|V|$. After determinization, the deterministic parity automaton for W has size $2^{\mathcal{O}(|V|(|V|+k|\mathcal{M}|)\log(|V|+k|M|))} = 2^{\mathcal{O}((|V|+k|\mathcal{M}|)^3)}$.

Combining this with Theorem 8, we can conclude that the set of winning strategies of Minimizer in the unboundedness game can be recognized by a non-deterministic parity tree automaton of size

$$2^{\mathcal{O}\left(\left(2^{\mathcal{O}((|V|+k|\mathcal{M}|)^3)}\right)^9+|V|^3\right)} = 2\mathrm{EXP}\left((|V|+k|\mathcal{M}|)^3\right). \tag{1}$$

What remains to be shown is the connection between the value of $\mathcal{G}$ and the existence of a winning strategy of Minimizer in $\mathcal{G}^u$, i.e., that the set of winning strategies of Minimizer in $\mathcal{G}^u$ satisfies the conditions from Theorem 2. We will use Ramsey's theorem for a finite path π in $\mathcal{G}^u$ or $\mathcal{G}$ played consistently with a strategy using memory of size K_0. We write π as a sequence of vertices and memory states with edges labeled with the corresponding marks:

$$\pi = (v_0, q_0) \xrightarrow{m_0} (v_1, q_1) \xrightarrow{m_1} (v_2, q_2) \cdots \xrightarrow{m_{n-2}} (v_{n-1}, q_{n-1}),$$

where each v_i is a vertex and each q_i is a memory state (and $q_{i+1} = \mathrm{update}(q_i, v_i)$ according to the strategy). The path π induces a complete edge-colored undirected graph over $[n]$, in which an edge i, j is colored by (m, v_i, q_i, v_j, q_j), where m is the composition of the marks $m_i \circ m_{i+1} \circ \cdots \circ m_{j-1}$. Let l be the number of such colors for $\mathcal{G}^u$ and memory size K_0:

$$l = |\mathcal{M}| \cdot |V_u|^2 \cdot K_0^2 = |\mathcal{M}| \cdot K_0^2 \cdot (|V_m| \cdot |V_m|)^2 = |\mathcal{M}| \cdot K_0^2 \cdot (|V| \cdot 2^k)^4.$$

We write $\mathcal{R} = R(\underbrace{3, 3, \cdots, 3}_{l \text{ times}})$ for the Ramsey-number for 3-cliques with l colors. As $\mathcal{R} \leq 3l!$ [13], we get that $\mathcal{R} = 2^{\mathcal{O}(l)}$.

Recall that a mark m is idempotent if $m = m \circ m$. The following lemma (see [2]) is used in the next proof. To simplify notation, we write $i \subseteqq m(c^{>0}, j)$ if $i \in m(c^{>0}, j)$ or $i = m(c^{>0}, j)$.

Lemma 9. *Let m be an idempotent mark. Then, for all initial values c, and all $i < k$: if $i \not\subseteqq m(c^{>0}, i)$, then i will not appear in any $m(c^{>0}, j)$.*

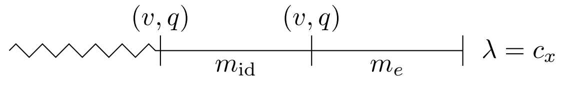
In the following, we show that, if g is a winning strategy of Minimizer in $\mathcal{G}^u$ with memory M, then $\mathrm{val}_g\, \mathcal{G}(v)$ is bounded.

Proposition 10. *Let g be a strategy of Minimizer winning in $\mathcal{G}^u$ from v and using memory M. Then, $\sup_{f \in \Sigma_{\max}} p(\alpha_{f,g}(v)) < 2\mathrm{EXP}(M^2 \cdot (|V| \cdot 2^k)^4)$.*

Proof. Let g be such an M-memory winning strategy. Consider the set of paths of length at most $\mathcal{R}$. We fix K as the maximal counter value plus 1 occurring anywhere on these paths when starting with initial counter values $c = (a, \cdots, a)$, where a is the maximal number occurring in any update function's matrices A or B. A rough upper bound can be computed as follows: after one application of any of the update functions, the maximal value is at most $a \cdot a \cdot k$ (the sum of all counters initialized with a, each weighted with a). After two steps, we get at most $k \cdot a \cdot k \cdot a \cdot a = k^2 a^{2+1}$. After $\mathcal{R}$ steps, we thus get $K \leq k^{\mathcal{R}} a^{\mathcal{R}+1} + 1 = 2^{\mathcal{O}(\mathcal{R})}$, and by the approximation of $\mathcal{R}$ above, $K \leq 2\mathrm{EXP}(M^2 \cdot (|V| \cdot 2^k)^4)$.

Note that, because of imperfect information and imperfect recall, g can also be viewed as a strategy for $\mathcal{G}$. Consider thus, towards a contradiction, a play α in $\mathcal{G}$ that is consistent with g and that has a payoff $\geq K$. Let further β be the corresponding play in $\mathcal{G}^u$ in which Maximizer never takes a Call, i.e., β consists only of a main part. We distinguish two cases: if α is infinite, then so is β. Because g is a winning strategy for Minimizer in $\mathcal{G}^u$, β – and thus α – violates the parity condition. But then the payoff for α is $-\infty$, a contradiction. If α is finite but has a payoff $\geq K$, we first observe the following (proved in [2]).

Claim. Every play consistent with g and with payoff $\geq K$ has a suffix $(*)$:



with m_{id} idempotent, such that for some j with $j \leqq m_{\mathrm{id}} \circ m_e(c_x)$: $j \in m_{\mathrm{id}}(j)$.

By the above claim, it follows that α contains a cycle that can be repeated arbitrarily many times by Maximizer. As repeating the cycle increases the payoff, repeating the cycle, taking a Call towards the Return, then repeating the cycle and taking the Call again, and so on, is a witness for a win of Maximizer in $\mathcal{G}^u$. Because of imperfect information and imperfect recall, this witness is consistent with g, contradicting the assumption that g is winning for Minimizer. $\square$

To prove the other item in Theorem 2, we show that Maximizer can achieve arbitrarily high payoffs against non-winning strategies of Minimizer in $\mathcal{G}^u$.

Proposition 11. *For every strategy g of Minimizer in $\mathcal{G}^u$ that is not winning from v,* $\mathrm{val}_g\, \mathcal{G}(v) = \infty$.

Proof. Let g be an arbitrary strategy of Minimizer that is not winning from v in $\mathcal{G}^u$. This means that there exists a consistent play $\alpha(g)$ won by Maximizer. Note that $\mathcal{G}^u$ is not necessarily determined, but $\mathcal{G}$ is determined (cf. Corollary 1). Thus, it suffices to show that, for every natural number $N \in \mathbb{N}$, Maximizer has a strategy to ensure a payoff $> N$ against g in $\mathcal{G}$. Recall that strategies of Minimizer in $\mathcal{G}$ correspond to strategies in $\mathcal{G}^u$. Let thus g and N be given. Maximizer can play as follows: play as in $\alpha(g)$ until the first Call occurs. Skip the Call-Return sequence. If $\alpha(g)$ is won via the parity condition, do this infinitely often. Otherwise, wait until the winning counter d has reached a value $> N$ and a Call occurs. Take the Call and realize the payoff as required. $\square$

We can now prove Theorem 2. Indeed, let Σ be the set of strategies winning for Minimizer in $\mathcal{G}^u$. By Equation 1 it is recognized by an automaton of the claimed size (with $m = |\mathcal{M}|$). Then, by Proposition 11, the first item of Theorem 2, holds, and by Proposition 10 the second item is true.

6 Outlook

The presented algorithm allows to solve a general class of counter parity games, which capture several cost models for dynamic systems. These cost models are also used in many online algorithms, thus it may be beneficial to apply affine counter parity games in this domain, in the spirit of [1]. A crucial part in our proof is played by games with imperfect information and imperfect recall – a class which has been studied in classical game theory, but so far received little attention in computer science. This motivates a future more systematic study of ω-regular games with imperfect recall.

References

1. B. Aminof, O. Kupferman, and R. Lampert. Formal analysis of online algorithms. In *Proc. of ATVA'11*, pages 213–227, 2011.
2. D. Berwanger, Ł. Kaiser, and S. Leßenich. Imperfect recall and counter games. Research Report LSV-11-20, LSV, ENS Cachan, France, 2011. Available under http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2011-20.pdf.
3. T. Brázdil, V. Forejt, J. Krcál, J. Kretínský, and A. Kucera. Continuous-time stochastic games with time-bounded reachability. In *Proc. of FSTTCS'09*, pages 61–72, 2009.
4. K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Generalized mean-payoff and energy games. In *Proc. of FSTTCS'10*, pages 505–516, 2010.
5. K. Chatterjee, T. A. Henzinger, and F. Horn. The complexity of request-response games. In *Proc. of LATA'11*, pages 227–237, 2011.
6. L. de Alfaro, M. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *Theoretical Computer Science*, 345(1):139–170, 2005.
7. D. Fischer, E. Grädel, and Ł. Kaiser. Model checking games for the quantitative μ-calculus. *Theory Comput. Syst.*, 47(3):696–719, 2010.
8. D. Fischer and Ł. Kaiser. Model checking the quantitative μ-calculus on linear hybrid systems. In *Proc. of ICALP'11 (2)*, volume 6756 of *LNCS*, pages 404–415. Springer, 2011.
9. F. Horn, W. Thomas, and N. Wallmeier. Optimal strategy synthesis in request-response games. In *Proc. of ATVA'08*, pages 361–373, 2008.
10. Ł. Kaiser and S. Leßenich. Counting μ-calculus on structured transition systems. In *Proc. of CSL'12*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2012. To appear.
11. O. Kupferman, N. Piterman, and M. Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.
12. W. Thomas. Infinite games and verification. In *Proc. of CAV'02,*, volume 2404 of *LNCS*, pages 58–64. Springer, 2002.
13. H. Wan. Upper bounds for Ramsey numbers R(3, 3,. . ., 3) and Schur numbers. *Journal of Graph Theory*, 26(3):119–122, 1997.
14. M. Zimmermann. Optimal bounds in parametric LTL games. In *Proc. of GandALF'11*, pages 146–161, 2011.