Robust Exponential Worst Cases for Divide-et-Impera Algorithms for Parity Games

Massimo Benerecetti & Daniele Dell'Erba
Università degli Studi di Napoli Federico II

Fabio Mogavero
Università degli Studi di Verona

The McNaughton-Zielonka *divide et impera* algorithm is the simplest and most flexible approach available in the literature for determining the winner in a parity game. Despite its theoretical exponential worst-case complexity and the negative reputation as a poorly effective algorithm in practice, it has been shown to rank among the best techniques for the solution of such games. Also, it proved to be resistant to a lower bound attack, even more than the strategy improvements approaches, and only recently a family of games on which the algorithm requires exponential time has been provided by Friedmann. An easy analysis of this family shows that a simple memoization technique can help the algorithm solve the family in polynomial time. The same result can also be achieved by exploiting an approach based on the dominion-decomposition techniques proposed in the literature. These observations raise the question whether a suitable combination of dynamic programming and game-decomposition techniques can improve on the exponential worst case of the original algorithm. In this paper we answer this question negatively, by providing a robustly exponential worst case, showing that no possible intertwining of the above mentioned techniques can help mitigating the exponential nature of the *divide et impera* approaches.

1 Introduction

Parity games [38] are perfect-information two-player turn-based games of infinite duration, usually played on finite directed graphs. Their vertices, labeled by natural numbers called *priorities*, are assigned to one of two players, named *Even* and *Odd* or, simply, 0 and 1, respectively. A play in the game is an infinite sequence of moves between vertices and it is said to be winning for player 0 (*resp.*, 1), if the maximal priority encountered infinitely often along the play is *even* (*resp.*, odd). These games have been extensively studied in the attempt to find efficient solutions to the problem of determining the winner. From a complexity theoretic perspective, this decision problem lies in NPTIME $\cap$ CONPTIME [16], since it is *memoryless determined* [15, 30, 31, 38]. It has been even proved to belong to UPTIME $\cap$ COUPTIME [23] and, very recently, to be solvable in quasi-polynomial time [11]. They are the simplest class of games in a wider family with similar complexities and containing, *e.g.*, *mean payoff games* [14,22], *discounted payoff games* [47], and *simple stochastic games* [13]. In fact, polynomial time reductions exist from parity games to the latter ones. However, despite being the most likely class among those games to admit a polynomial-time solution, the answer to the question whether such a solution exists still remains elusive. The effort devoted to provide efficient solutions stems primarily from the fact that many problems in formal verification and synthesis can be reformulated in terms of solving parity games. Emerson, Jutla, and Sistla [16] have shown that computing winning strategies for these games is linear-time equivalent to solving the modal μCALCULUS model checking problem [17]. Parity games also play a crucial role in automata theory [15,29,37], where they can be applied to solve the complementation problem for alternating automata [21] and the emptiness of the corresponding nondeterministic tree automata [29]. These automata, in turn, can be used to solve the satisfiability and model checking problems for expressive logics, such as the modal [45] and alternating [1,43] μCALCULUS, ATL* [1,42],

P. Bouyer, A. Orlandini & P. San Pietro (Eds.): 8th Symposium on Games, Automata, Logics and Formal Verification (GandALF'17)
EPTCS 256, 2017, pp. 121–135, doi:10.4204/EPTCS.256.9

Strategy Logic [12, 33–36], Substructure Temporal Logic [8, 9], and fixed-point extensions of guarded first-order logics [10].

Previous exponential solutions essentially divide into two families. The first one collects procedures that attempt to directly build winning strategies for the two players on the entire game. To such family belongs the *Small Progress Measure* approach by Jurdziński [24], which exploits the connection between the notions of progress measures [28] and winning strategies. A second approach in same vein is the *Strategy Improvement* algorithm by Jurdziński and Vöge [44], based on the idea of iteratively improving an initial, non necessarily winning, strategy.

The second family gathers, instead, the approaches based on decomposing the solution of a game into the analysis of its subgames. To this family belong the so called *divide et impera* approaches led by the *Recursive* algorithm proposed by Zielonka [46], which adapts to parity games an earlier algorithm proposed by McNaughton for Muller games [32]. Intuitively, it decomposes the input game into subgames and solves them recursively. Using the Recursive algorithm as a back-end, and in the attempt to obtain a better upper bound, the *Dominion Decomposition* [26, 27] and the *Big Step* [41] approaches were devised. Both share the idea of intertwining the recursive calls of the back-end with a preprocessing phase, applied to the current subgame, in search of a sufficiently small dominion for some player $\wp$, *i.e.*, a set of positions from where $\wp$ wins without ever exiting the set. The first technique does so by means of a brute force search, while the second one exploits a suitable variation of the Small Progress Measure procedure. A different direction has been followed recently within the decomposition-based family, that leads to a novel solution technique based on the notion of *priority promotion* [5–7]. The approach relies on a new procedure that finds dominions of arbitrary size, which proved to be quite efficient in practice and exhibits the best space complexity among the known solution algorithms, even better than the recently introduced quasi-linear space algorithms [18, 25].

The literature also suggests several heuristics to tune parity game solvers. One of the most successful ones is that of decomposing the game into strongly-connected components (SCCs, for short) and solving it SCC-wise. *SCC-decomposition*, together with some other minor techniques such as removal of self-cycles and priority compression, can significantly improve the solution process, as empirically demonstrated in [20]. The same authors also show that, against the negative reputation as far as performances are concerned, the Recursive algorithm often stands out as the best solver among those proposed in the literature, particularly when paired with the SCC-decomposition heuristic. Despite having a quite straightforward exponential upper bound, this algorithm has resisted an exponential lower bound for more than ten years, until Friedmann [19] devised an indexed family of games that forces the algorithm to execute a number of recursive calls that grows exponentially with the index. The family is also resilient to the SCC-decomposition technique, since each subgame passed to a recursive call always forms a single SCC. On a closer look, however, the games proposed there force an exponential behavior by requiring the algorithm to repeatedly solve a small number of subgames, actually only a linear number of them. As a consequence, all those games are amenable to a polynomial-time solution, by simply providing the algorithm with a suitable memoization mechanism that prevents it from wasting computational resources on solving already solved subgames. For different reasons, also a dominion decomposition approach can break the lower bound easily, as most of the subgames of a game in the family contain a dominion of constant size.

These observations raise the question whether the Recursive algorithm admits an exponential lower bound robust enough to be resilient to a suitable intertwining with memoization, SCC-decomposition, and dominion decomposition techniques. The difficulty here is that such a robust worst case should induce an exponential number of different subgames to prevent memoization from being of any help. At the same time, each of those subgames must contain a single SCC and only dominions of sufficiently large size to

prevent both SCC-decomposition and dominion decomposition techniques from simplifying the game. In this paper, we answer positively to the question, by providing a robust, and harder, worst case family that meets all the above requirements, thereby shading some light on the actual power of aforementioned techniques and sanctioning that no combination of them can indeed help improving the exponential lower bound of the *divide et impera* approaches.

A recent breakthrough [11] by Calude *et al.* proposes a succinct reduction from parity to reachability games based on a clever encoding of the sequences of priorities a player finds along a play. This allows for a mere quasi-polynomial blow up in the size of the underlying graph and sets the basis of the fixed-parameter tractability *w.r.t.* the number of priorities. The approach has been then considerably refined in [18], where these encodings are modeled as progress measures. A similar technique is also used in [25]. Despite the theoretical relevance of this new idea, preliminary experiments [4] seem to suggest that the practical impact of the result does not match the theoretical one, as all exponential algorithms outperform, often by orders of magnitude, the current implementations of the quasi-polynomial ones, which do not scale beyond few hundred vertices. This evaluation is consistent with the fact that the new techniques essentially amount to clever and succinct encodings embedded within a brute force search, which makes matching quasi-polynomial worst cases quite easy to find. These observations suggest that the road to a polynomial solution may need to take another direction. Our work is, therefore, intended to evaluate the weaknesses of classic exponential algorithms, in the same vein of [39, 40], where the authors study the pitfalls of existing exponential algorithms for graphs isomorphism, in spite of the fact that a quasi-polynomial, but impractical, algorithm exists [3]. We believe that a better understanding of the different issues of the known approaches may lead to progress in the quest for a polynomial algorithm.

2 Parity Games

Let us first briefly recall the notation and basic definitions concerning parity games that expert readers can simply skip. We refer to [2] [46] for a comprehensive presentation of the subject.

A two-player turn-based *arena* is a tuple $\mathscr{A} = \langle \mathrm{Ps}^0, \mathrm{Ps}^1, Mv \rangle$, with $\mathrm{Ps}^0 \cap \mathrm{Ps}^1 = \emptyset$ and $\mathrm{Ps} \triangleq \mathrm{Ps}^0 \cup \mathrm{Ps}^1$, such that $\langle \mathrm{Ps}, Mv \rangle$ is a finite directed graph. Ps^0 (*resp.*, Ps^1) is the set of positions of player 0 (*resp.*, 1) and $Mv \subseteq \mathrm{Ps} \times \mathrm{Ps}$ is a left-total relation describing all possible moves. A *path* in $\mathrm{V} \subseteq \mathrm{Ps}$ is an infinite sequence $\pi \in \mathrm{Pth}(\mathrm{V})$ of positions in V compatible with the move relation, *i.e.*, $(\pi_i, \pi_{i+1}) \in Mv$, for all $i \in \mathbb{N}$. A positional *strategy* for player $\wp \in \{0,1\}$ on $\mathrm{V} \subseteq \mathrm{Ps}$ is a function $\sigma_\wp \in \mathrm{Str}^\wp(\mathrm{V}) \subseteq (\mathrm{V} \cap \mathrm{Ps}^\wp) \to \mathrm{V}$, mapping each $\wp$-position $v \in \mathrm{V} \cap \mathrm{Ps}^\wp$ to position $\sigma_\wp(v) \in \mathrm{V}$ compatible with the move relation, *i.e.*, $(v, \sigma_\wp(v)) \in Mv$. By $\mathrm{Str}^\wp(\mathrm{V})$ we denote the set of all $\wp$-strategies on V. A *play* in $\mathrm{V} \subseteq \mathrm{Ps}$ from a position $v \in \mathrm{V}$ *w.r.t.* a pair of strategies $(\sigma_0, \sigma_1) \in \mathrm{Str}^0(\mathrm{V}) \times \mathrm{Str}^1(\mathrm{V})$, called $((\sigma_0, \sigma_1), v)$-*play*, is a path $\pi \in \mathrm{Pth}(\mathrm{V})$ such that $\pi_0 = v$ and, for all $i \in \mathbb{N}$, if $\pi_i \in \mathrm{Ps}^0$, then $\pi_{i+1} = \sigma^0(\pi_i)$ else $\pi_{i+1} = \sigma^1(\pi_i)$.

A *parity game* is a tuple $\mathcal{D} = \langle \mathscr{A}, \mathrm{Pr}, \mathsf{pr} \rangle$, where $\mathscr{A}$ is an arena, $\mathrm{Pr} \subset \mathbb{N}$ is a finite set of priorities, and $\mathsf{pr} : \mathrm{Ps} \to \mathrm{Pr}$ is a *priority function* assigning a priority to each position. The priority function can be naturally extended to games and paths as follows: $\mathsf{pr}(\mathcal{D}) \triangleq \max_{v \in \mathrm{Ps}} \mathsf{pr}(v)$; for a path $\pi \in \mathrm{Pth}$, we set $\mathsf{pr}(\pi) \triangleq \limsup_{i \in \mathbb{N}} \mathsf{pr}(\pi_i)$. A set of positions $\mathrm{V} \subseteq \mathrm{Ps}$ is a $\wp$-*dominion*, with $\wp \in \{0,1\}$, if there exists a $\wp$-strategy $\sigma_\wp \in \mathrm{Str}^\wp(\mathrm{V})$ such that, for all $\overline{\wp}$-strategies $\sigma_{\overline{\wp}} \in \mathrm{Str}^{\overline{\wp}}(\mathrm{V})$ and positions $v \in \mathrm{V}$, the induced $((\sigma_0, \sigma_1), v)$-play π has priority of parity $\wp$, *i.e.*, $\mathsf{pr}(\pi) \equiv_2 \wp$. In other words, $\sigma_\wp$ only induces on V plays whose maximal priority visited infinitely often has parity $\wp$. The *winning region* for player $\wp \in \{0,1\}$ in game $\mathcal{D}$, denoted by $\mathrm{Wn}^\wp_{\mathcal{D}}$, is the maximal set of positions that is also a $\wp$-dominion in $\mathcal{D}$. Since parity games are determined games [15], meaning that from each position one of the two players wins, the two winning regions of a game $\mathcal{D}$ form a partition of its positions, *i.e.*, $\mathrm{Wn}^0_{\mathcal{D}} \cup \mathrm{Wn}^1_{\mathcal{D}} = \mathrm{Ps}_{\mathcal{D}}$. By $\mathcal{D} \setminus \mathrm{V}$

we denote the maximal subgame of $\mathcal{D}$ with set of positions Ps' contained in $\mathrm{Ps}\setminus\mathrm{V}$ and move relation Mv' equal to the restriction of Mv to Ps'. The $\wp$-predecessor of V, in symbols $\mathsf{pre}^{\wp}(\mathrm{V}) \triangleq \{v \in \mathrm{Ps}^{\wp} : Mv(v) \cap \mathrm{V} \neq \emptyset\} \cup \{v \in \mathrm{Ps}^{\overline{\wp}} : Mv(v) \subseteq \mathrm{V}\}$, collects the positions from which player $\wp$ can force the game to reach some position in V with a single move. The $\wp$-attractor $\mathsf{atr}^{\wp}(\mathrm{V})$ generalizes the notion of $\wp$-predecessor $\mathsf{pre}^{\wp}(\mathrm{V})$ to an arbitrary number of moves. Thus, it corresponds to the least fix-point of that operator. When $\mathrm{V} = \mathsf{pre}^{\wp}(\mathrm{V})$, player $\wp$ cannot force any position outside V to enter this set. For such a V, the set of positions of the subgame $\mathcal{D} \setminus \mathrm{V}$ is precisely $\mathrm{Ps} \setminus \mathrm{V}$. When confusion cannot arise, we may abuse the notation and write $\mathcal{D}$ to mean its set of positions $\mathrm{Ps}_{\mathcal{D}}$.

3 The Recursive Algorithm

Algorithm 1: Recursive algorithm.

```
   signature sol: PG →_D 2^{Ps_D} × 2^{Ps_D}
   function sol(D)
1      (D_L, ℘) ← f_L(D)
2      (Wn^0_L, Wn^1_L) ← sol(D_L)
3      if pre^{℘̄}_D(Wn^{℘̄}_L) \ Wn^{℘̄}_L = ∅ then
4          (Wn^{℘}, Wn^{℘̄}) ← (Ps_D \ Wn^{℘̄}_L, Wn^{℘̄}_L)
       else
5          D_R ← f_R(D, Wn^{℘̄}_L, ℘̄)
6          (Wn^0_R, Wn^1_R) ← sol(D_R)
7          (Wn^{℘}, Wn^{℘̄}) ← (Wn^{℘}_R, Ps_D \ Wn^{℘}_R)
8      return (Wn^0, Wn^1)
```

Algorithm 2: Left-subgame function.

```
   signature f_L: PG → PG × {0,1}
   function f_L(D)
1      ℘ ← pr(D) mod 2
2      D* ← D \ atr^{℘}_D(pr_D^{-1}(pr(D)))
3      return (D*, ℘)
```

Algorithm 3: Right-subgame function.

```
   signature f_R: PG ×_D 2^{Ps_D} × {0,1} → PG
   function f_R(D, W, ℘)
1      D* ← D \ atr^{℘}_D(W)
2      return D*
```

The Recursive procedure, reported in Algorithm 1 and proposed by Zielonka [46] in an equivalent version, solves a parity game $\mathcal{D}$ by decomposing it into two subgames, each of which is, then, solved recursively. Intuitively, the procedure works as follows. Algorithm 1, by means of Algorithm 2, starts by collecting all the positions that are forced to pass through a position with maximal priority $p \triangleq \mathsf{pr}(\mathcal{D})$ in that game. This first step results in computing the set $\mathrm{A} \triangleq \mathsf{atr}^{\wp}_{\mathcal{D}}(\mathsf{pr}_{\mathcal{D}}^{-1}(p))$, *i.e.*, the attractor to the set $\mathsf{pr}_{\mathcal{D}}^{-1}(p)$ of positions with priority p *w.r.t.* player $\wp \triangleq p \bmod 2$. The subgame $\mathcal{D}_{\mathrm{L}}$ is, then, obtained from $\mathcal{D}$ by removing A from it and solved recursively. The result is a partitioning of the positions of $\mathcal{D}_{\mathrm{L}}$ into two winning regions, $\mathrm{Wn}^0_{\mathrm{L}}$ and $\mathrm{Wn}^1_{\mathrm{L}}$, one per player. At this point, the algorithm checks whether the subgame $\mathcal{D}_{\mathrm{L}}$ is completely won by $\wp$ or, more generally, if the adversary $\overline{\wp}$ cannot force any other position in $\mathcal{D}$ into its own winning region $\mathrm{Wn}^{\overline{\wp}}_{\mathrm{L}}$ in one move. In other words, none of the winning positions of the adversary $\overline{\wp}$ can attract something outside that region, *i.e.*, $\mathsf{pre}^{\overline{\wp}}_{\mathcal{D}}(\mathrm{Wn}^{\overline{\wp}}_{\mathrm{L}}) \setminus \mathrm{Wn}^{\overline{\wp}}_{\mathrm{L}} = \emptyset$. If this is the case, the entire game $\mathcal{D}$ is solved. Indeed, the positions of $\mathcal{D}$ winning for $\wp$ are all its positions except, possibly, for those won by $\overline{\wp}$ in subgame $\mathcal{D}_{\mathrm{L}}$ (see Line 4 of Algorithm 1). If, on the other hand, the above condition does not hold, the winning region of $\overline{\wp}$ can be extended with some other positions in $\mathcal{D}$. Let $\mathrm{B} \triangleq \mathsf{atr}^{\overline{\wp}}_{\mathcal{D}}(\mathrm{Wn}^{\overline{\wp}}_{\mathrm{L}})$ be the set collecting all such positions. Observe that all the positions in B are certainly winning for $\overline{\wp}$ in the entire game, as, from each such position, $\overline{\wp}$ can force entering its own winning region $\mathrm{Wn}^{\overline{\wp}}_{\mathrm{L}}$, from which its opponent $\wp$ cannot escape. The residual subgame $\mathcal{D}_{\mathrm{R}}$, obtained by removing B form $\mathcal{D}$, as computed by Algorithm 3, may now contain positions winning for either player, and, therefore, still needs to be solved recursively (see Line 6 of Algorithm 1). All the positions of $\mathcal{D}_{\mathrm{R}}$ that turn out to be winning for $\wp$ in that

game, namely $\mathrm{Wn}_{\mathrm{R}}^{\wp}$, are, then, all and only those positions winning for $\wp$ in the entire game $\mathcal{D}$, while the remaining ones are winning for $\overline{\wp}$ (see Line 7 of Algorithm 1).

As shown by Friedmann in [19], the algorithm admits a worst case family of games $\{\mathcal{D}^k\}_{k=1}^{\omega}$ that requires a number of recursive calls exponential in k. The reason is essentially the following. Each game $\mathcal{D}^k$ of that family contains all $\mathcal{D}^j$, with $1 \leq j < k$, as subgames. Each recursive call that receives as input one such subgame $\mathcal{D}^j$ requires to eventually solve both $\mathcal{D}^{j-1}$ and $\mathcal{D}^{j-2}$. As a consequence, the number of recursive calls performed by the algorithm on game $\mathcal{D}^k$ can be put in correspondence with a Fibonacci sequence. This proves that their number grows at least as fast as the sequence of the Fibonacci numbers, namely that their number is $\Omega\Big(((1+\sqrt{5})/2)^k\Big)$. The very reason that makes this family exponential also makes it amenable to a polynomial-time solution. It suffices to endow the Recursive algorithm with a memoization mechanism that, for each solved game $\mathcal{D}$, records the triple $(\mathcal{D}, \mathrm{Wn}_{\mathcal{D}}^{0}, \mathrm{Wn}_{\mathcal{D}}^{1})$. Each recursive call can, then, directly extract the winning regions of a subgame that is already contained in the collection, thus preventing the procedure from solving any subgame more than once. Not only does the resulting procedure make Friedman worst case vain, but it also speeds up the solution of games significantly, as long as the number of repeated subgames remains relatively small, *e.g.*, linear in the size of the original game, which is often the case in practice.

4 Memoization Resilient Games

As mentioned above, the main requirement for an exponential worst case family for the memoized version of the Recursive algorithm is to contain games that force the procedure to solve an exponential number of different subgames. In this section we shall focus primarily on this problem, by showing that such a family exists. More generally, we identify a *core family* $\{\mathcal{D}_{\mathsf{C}}^k\}_{k=1}^{\omega}$ of games enjoying that specific property. For each $k \in \mathbb{N}_+$, game $\mathcal{D}_{\mathsf{C}}^k$ contains $2k+1$ gadgets, each one formed by three positions α_i, β_i, and γ_i, for $i \in [0, 2k]$. The positions β_i and γ_i, in gadget i, share the same priority i and opposite owners, namely player i mod 2 for β_i and $(i+1)$ mod 2 for γ_i. The position α_i in the gadget has the same owner as the corresponding β_i. These positions are leading ones, having higher priorities than all the β's and γ's of the other gadgets. Positions within gadget i are connected as follows: α_i can only move to β_i; β_i can only move to γ_i; γ_i can choose either to move to β_i or to stay in γ_i itself. Two adjacent gadgets, of indexes i and $i+1$, are connected by only two moves: one from γ_i to α_{i+1} and one from β_{i+1} to α_i. Figure 1 depicts game $\mathcal{D}_{\mathsf{C}}^2$. The gray portion in the figure represents game $\mathcal{D}_{\mathsf{C}}^1$, where all the priorities of all its α's have been increased by 2, so as to comply with the requirement that the α's have the higher priorities. Indeed, in general, given an index k, game $\mathcal{D}_{\mathsf{C}}^{k+1}$ is obtained from $\mathcal{D}_{\mathsf{C}}^k$ by increasing by two units the priorities of each α_i in the gadgets of $\mathcal{D}_{\mathsf{C}}^k$ and adding two new gadgets with indexes $2k+1$ and $2(k+1)$ connected as in figure. The games in the core family are formally described by the following definition.

Definition 4.1 (Core Family)**.** *The* core family $\{\mathcal{D}_{\mathsf{C}}^k\}_{k=1}^{\omega}$*, where* $\mathcal{D}_{\mathsf{C}}^k \triangleq \langle \mathscr{A}, \mathrm{Pr}, \mathsf{pr} \rangle$*,* $\mathscr{A} \triangleq \langle \mathrm{Ps}^0, \mathrm{Ps}^1, Mv \rangle$*, and* $\mathrm{Pr} \triangleq [0, 4k]$*, is defined as follows. For any index* $k \geq 1$*, the set of positions* $\mathrm{Ps} \triangleq \{\alpha_i, \beta_i, \gamma_i : 0 \leq i \leq 2k\}$ *of* $\mathcal{D}_{\mathsf{C}}^k$ *is divided into three categories:*

- α_i *belongs to player* $\wp \triangleq i$ mod 2, i.e., $\alpha_i \in \mathrm{Ps}^{\wp}$*, and has priority* $\mathsf{pr}(\alpha_i) \triangleq 2k+i+1$*;*
- β_i *belongs to player* $\wp \triangleq i$ mod 2, i.e., $\beta_i \in \mathrm{Ps}^{\wp}$*, and has priority* $\mathsf{pr}(\beta_i) \triangleq i$*;*
- γ_i *belongs to player* $\overline{\wp} \triangleq (i+1)$ mod 2, i.e., $\gamma_i \in \mathrm{Ps}^{\overline{\wp}}$*, and has priority* $\mathsf{pr}(\gamma_i) \triangleq i$*.*

Moreover, the moves from positions α_i*,* β_i*, and* γ_i*, with* $0 \leq i \leq 2k$*, are prescribed as follows:*

- α_i *has a unique move to* β_i*,* i.e., $Mv(\alpha_i) = \{\beta_i\}$*;*

- β_i *has one move to* γ_i *and one to* α_{i-1}*, if* $i > 0$, i.e., $Mv(\beta_i) = \{\gamma_i\} \cup \{\alpha_{i-1} : i > 0\}$*;*
- γ_i *has one move to* γ_i *itself, one to* β_i*, and, if* $i < 2k$*, one to* α_{i+1}, i.e., $Mv(\gamma_i) = \{\beta_i, \gamma_i\} \cup \{\alpha_{i+1} : i < 2k\}$.

It is not hard to verify that, for any $k \in \mathbb{N}_+$, the game $\mathfrak{D}^k_{\mathsf{C}}$ is completely won by player 0 and contains precisely $6k+3$ positions and $12k+4$ moves. As we shall see later in detail, the solution of each such games requires Algorithm 1 to solve an exponential number of different subgames.

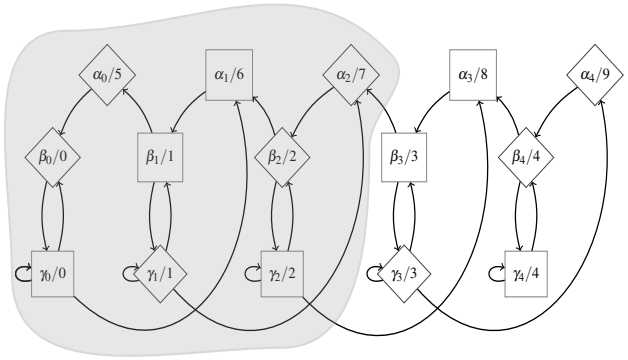

Figure 1: Game $\mathfrak{D}^2_{\mathsf{C}}$ of the core family.

These core games form the backbone for a more general framework, consisting of an entire class of game families, with the property that each of them remains resilient to memoization techniques. Essentially, each game in any such family extends a core game. In order to define such a wider class, let us first establish what counts as a suitable extension of a core. Clearly, for a game $\mathfrak{D}$ to be an extension of $\mathfrak{D}^k_{\mathsf{C}}$, for some index $k \in \mathbb{N}_+$, it must contain $\mathfrak{D}^k_{\mathsf{C}}$ as a subgame. However, in order to prevent the Recursive algorithm from disrupting the core while processing $\mathfrak{D}$, we have to enforce some additional requirements. In particular, we need the algorithm to behave on $\mathfrak{D}$ virtually in the same way as it does on the core subgame. To this end, we require that all positions α_i still have the maximal priorities as in the core. Moreover, all positions α_i and β_i cannot have additional moves in $\mathfrak{D}$ *w.r.t.* those contained in the core. Finally, if γ_i can escape to some position v outside the core, then v does not have a higher priority, it has a move back to γ_i, and belongs to the opponent player *w.r.t.* γ_i. This condition ensures that no γ_i can decide to escape the core without being bounced back immediately by the opponent. The following definition makes the notion of extension precise.

Definition 4.2 (Core Extension)**.** *An arbitrary parity game* $\mathfrak{D} \in \mathrm{PG}$ *is a* core extension *of* $\mathfrak{D}^k_{\mathsf{C}}$*, for a given index* $k \in \mathbb{N}_+$*, if the following four conditions hold:*

1. $\mathfrak{D} \setminus \mathrm{P} = \mathfrak{D}^k_{\mathsf{C}}$*, where* $\mathrm{P} \triangleq \{v \in \mathfrak{D} : v \notin \mathfrak{D}^k_{\mathsf{C}}\}$*;*
2. $\mathsf{pr}_{\mathfrak{D}}(v) < \mathsf{pr}_{\mathfrak{D}}(\alpha_0)$*, for all* $v \in \mathrm{P}$*;*
3. $\{\alpha_i, \beta_i \in \mathfrak{D} : 0 \leq i \leq 2k\} \cap (Mv(\mathrm{P}) \cup Mv^{-1}(\mathrm{P})) = \emptyset$*;*
4. $v \in \mathrm{Ps}^{\wp}$*,* $\gamma_i \in Mv(v)$*, and* $\mathsf{pr}(v) \leq i$*, for all* $v \in \mathrm{P} \cap Mv(\gamma_i)$*,* $i \in [0, 2k]$*, and* $\wp \triangleq i \bmod 2$.

We shall denote with $\mathrm{PG}_{\mathsf{C}} \subseteq \mathrm{PG}$ *the set of all core extensions of* $\mathfrak{D}^k_{\mathsf{C}}$*, for any index* $k \in \mathbb{N}_+$.

We can now define the abstract notion of worst-case family that extends the core family, while still preserving the same essential properties that we are going to prove shortly.

Definition 4.3 (Worst-Case Family)**.** *A family of parity games* $\{\mathfrak{D}^k\}_{k=1}^{\omega}$ *is a* worst-case family *if* $\mathfrak{D}^k$ *is a core extension of* $\mathfrak{D}^k_{\mathsf{C}}$*, for every index* $k \in \mathbb{N}_+$.

In order to prove that any worst-case family requires an exponential number of different subgames to be solved, we shall characterize a suitable subtree of the recursion tree generated by the algorithm, when called on one of the games in the family. Starting from the root, which contains the original game $\mathfrak{D}^k$, we fix specific observation points in the recursion tree that are identified by sequences in the set $w \in \{\mathtt{L}, \mathtt{R}\}^{\leq k}$, where $\mathtt{L}$ (*resp.*, $\mathtt{R}$) denotes the recursive call on the left (*resp.*, right) subgame. Each sequence w identifies two subgames, $\widehat{\mathfrak{D}}^k_w$ and $\mathfrak{D}^k_w$, of $\mathfrak{D}^k$ that correspond to the input subgames of two successive nested calls. In

the analysis of the recursion tree, we shall only take into account the left subgame $\mathcal{D}^k_w$ of each $\widehat{\mathcal{D}}^k_w$, thus disregarding its right subtree as it is inessential to the argument. An example of the resulting subgame tree for game $\mathcal{D}^1_{\mathtt{C}}$ of the core family is depicted in Figure 2. According to Algorithm 1 on input $\mathcal{D}^1_{\mathtt{C}} = \mathcal{D}^1_{\varepsilon}$, the first (left) recursive call is executed on the subgame obtained by removing the 1-attractor to the positions with maximal priority, in this case α_9, which only contains α_9. Therefore, the left subgame coincides precisely with $\widehat{\mathcal{D}}^1_{\mathtt{L}}$. The second call is executed on the game obtained by removing the 0-attractor in $\mathcal{D}^1_{\varepsilon}$ to the winning region for player 0 of the left subgame $\widehat{\mathcal{D}}^1_{\mathtt{L}}$. In this case, that winning region is precisely $\{\beta_2, \gamma_2\}$, and its 0-attractor is $\{\alpha_2, \gamma_1, \beta_2, \gamma_2\}$. As consequence, the subgame passed to the right-hand call precisely coincides with $\widehat{\mathcal{D}}^1_{\mathtt{R}}$. The rest of the subtree is generated applying the same reasoning. The following definition generalizes this notion to a game of any worst-case family and characterizes the portion of the recursion tree we are interested in analyzing.

Definition 4.4 (Induced Subgame Tree)**.** *Given a worst-case family* $\{\mathcal{D}^k\}_{k=1}^{\omega}$*, the* induced subgame tree $\mathbb{G}^k \triangleq \{\mathcal{D}^k_w\}_{w\in\{\mathtt{L},\mathtt{R}\}^{\leq k}} \cup \{\widehat{\mathcal{D}}^k_w\}^{w\neq\varepsilon}_{w\in\{\mathtt{L},\mathtt{R}\}^{\leq k+1}}$ w.r.t. *an index* $k \in \mathbb{N}_+$ *is defined inductively on the structure of the sequence* $w \in \{\mathtt{L},\mathtt{R}\}^{\leq k}$ *as follows, where* $\mathcal{D}^k_{\varepsilon} \triangleq \mathcal{D}^k$ *and* $z \triangleq 2(k-|w|)$*:*

1. $\widehat{\mathcal{D}}^k_{w\mathtt{L}} \triangleq \mathcal{D}^k_w \setminus \mathsf{atr}^1_{\mathcal{D}^k_w}(\{\alpha_z\})$*;*
2. $\widehat{\mathcal{D}}^k_{w\mathtt{R}} \triangleq \mathcal{D}^k_w \setminus \mathsf{atr}^{\mathrm{o}}_{\mathcal{D}^k_w}(\mathrm{Wn}^{\mathrm{o}}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}})$*;*
3. $\mathcal{D}^k_w \triangleq \widehat{\mathcal{D}}^k_w \setminus \mathsf{atr}^{\mathrm{o}}_{\widehat{\mathcal{D}}^k_w}(\{\alpha_{z+1}\})$*, if* $w \neq \varepsilon$*.*

Before proceeding with proving the main result of this section, we need some additional properties of the induced subgame tree of any worst-case family. The following lemma states some invariant of the elements contained in the tree of a games for $\mathcal{D}^k$, extending the core $\mathcal{D}^k_{\mathtt{C}}$, that will be essential to the result. In particular, they ensure that all of them are subgames of $\mathcal{D}^k$ and that, depending on the identifying sequence w, they contain the required leading positions α_i of the core. In addition, it states two important properties of every left child in the tree, *i.e.*, those elements identified by a sequence w ending with $\mathtt{L}$. Both of them will be instrumental in proving that all the subgames in the tree are indeed different and to assess their number, as we shall see in Lemmas 4.3 and 4.4. The first one ensures that each such game necessarily contain a specific position γ_i, with the index i depending on w. The second one characterizes the winning region for player 0 of the left-child subgames $\widehat{\mathcal{D}}^k_{w\mathtt{L}}$. It states that, in each such game, the winning positions for player 0 contained in the corresponding core $\mathcal{D}^k_{\mathtt{C}}$ are all its β_{2j} and γ_{2j}, with even index greater than the maximal index of a leading position α_x in that game. Indeed, as soon as the higher positions α_i, with $i \in [x+1, k]$, are removed from the game, each residual corresponding γ_{2j}, possibly together with its associated β_{2j}, is necessarily contained in an independent 0-dominion.

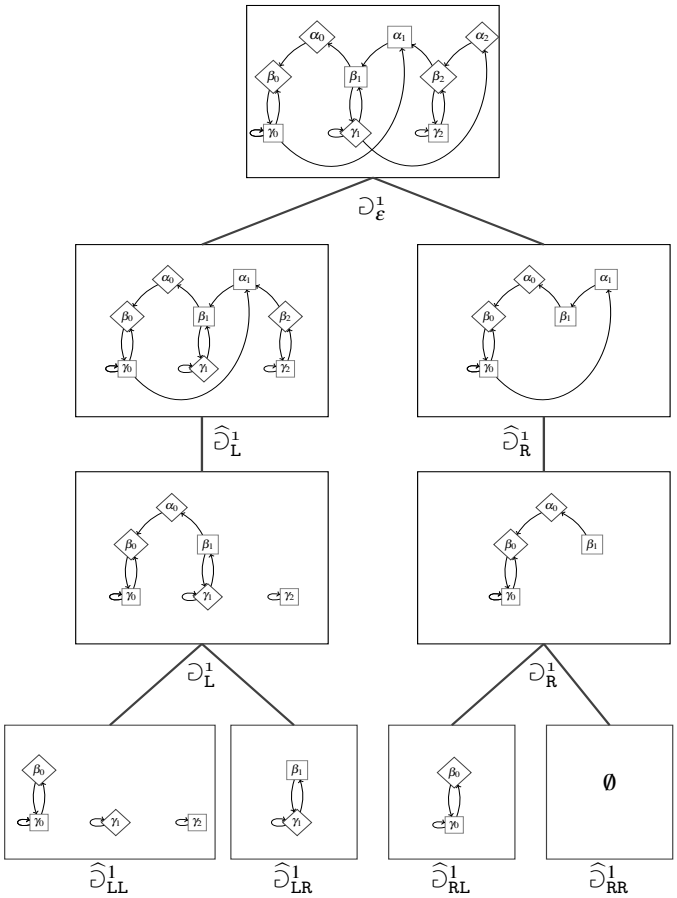

Figure 2: The induced subgame tree $\mathbb{G}^1$ of $\mathcal{D}^1$.

In the sequel, by $\mathsf{lst}(w)$ we denoted the last position of a non-empty sequence $w \in \{\mathtt{L},\mathtt{R}\}^+$.

Lemma 4.1. *For any index $k \in \mathbb{N}_+$ and sequence $w \in \{\mathtt{L},\mathtt{R}\}^{\leq k+1}$, let $z \triangleq 2(k-|w|)$. Then, the following properties hold, where $|w| \leq k$, in the first three items, and $w \neq \varepsilon$, in the remaining ones:*

1. *$\mathcal{D}^k_w$ is a subgame of $\mathcal{D}^k$;*
2. *$\alpha_j \in \mathcal{D}^k_w$ iff $j \in [0,z]$;*
3. *$\gamma_j \in \mathcal{D}^k_w$, for all $j \in [0,z]$, and $\gamma_{z+1} \in \mathcal{D}^k_w$, if $w \neq \varepsilon$ and $\mathsf{lst}(w) = \mathtt{L}$;*
4. *$\widehat{\mathcal{D}}^k_w$ is a subgame of $\mathcal{D}^k$;*
5. *$\alpha_j \in \widehat{\mathcal{D}}^k_w$ iff $j \in [0,z+1]$;*
6. *$\gamma_j \in \widehat{\mathcal{D}}^k_w$, for all $j \in [0,z]$, and $\gamma_{z+1} \in \widehat{\mathcal{D}}^k_w$, if $|w| \leq k$, and $\gamma_0 \in \widehat{\mathcal{D}}^k_w$, otherwise, when $\mathsf{lst}(w) = \mathtt{L}$;*
7. *$\mathrm{Wn}^{\mathsf{o}}_{\widehat{\mathcal{D}}^k_w} \cap \mathcal{D}^k_{\mathsf{C}} = \{\beta_{z+2j}, \gamma_{z+2j} \in \widehat{\mathcal{D}}^k_w : j \in [1,|w|]\}$, if $\mathsf{lst}(w) = \mathtt{L}$.*

Finally, the next lemma simply establishes that all the subgames contained in the induced subgame tree of Definition 4.4 are indeed generated by the Recursive algorithm when called with input game $\mathcal{D}^k$ of some worst-case family.

Lemma 4.2. *For any index $k \in \mathbb{N}_+$ and sequence $w \in \{\mathtt{L},\mathtt{R}\}^{\leq k}$, the following properties hold:*

1. *$\mathsf{f}_{\mathtt{L}}(\mathcal{D}^k_w) = (\widehat{\mathcal{D}}^k_{w\mathtt{L}}, 1)$;*
2. *$\mathsf{f}_{\mathtt{R}}(\mathcal{D}^k_w, \mathrm{Wn}^{\mathsf{o}}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}, 0) = \widehat{\mathcal{D}}^k_{w\mathtt{R}}$;*
3. *$\mathsf{f}_{\mathtt{L}}(\widehat{\mathcal{D}}^k_w) = (\mathcal{D}^k_w, 0)$, if $w \neq \varepsilon$.*

We are now ready for the main result of this section, namely that the induced subgame tree contains elements which are all different from each other and whose number is exponential in the index k. We split the result into two lemmas. The first one simply states that any subgame in the left subtree of some $\mathcal{D}^k_w$ is different from any other subgame in the right subtree. The idea is that for for any subgame in the tree, all subgames of its left subtree contain at least one position, a specific position γ_i with i depending on w, that is not contained in any subgame of its right subtree.

Lemma 4.3. *For all indexes $k \in \mathbb{N}_+$ and sequences $w,v \in \{\mathtt{L},\mathtt{R}\}^*$, with $\ell \triangleq |w| + |v| \leq k$ and $z \triangleq 2(k-|w|)-1$, the following properties hold:*

1. *$\gamma_z \in \mathcal{D}^k_{w\mathtt{L}v}$ and $\gamma_z \in \widehat{\mathcal{D}}^k_{w\mathtt{L}v}$, if $\ell < k$, and $\gamma_0 \in \widehat{\mathcal{D}}^k_{w\mathtt{L}v}$, otherwise;*
2. *$\gamma_z \notin \mathcal{D}^k_{w\mathtt{R}v}$ and $\gamma_z \notin \widehat{\mathcal{D}}^k_{w\mathtt{R}v}$, if $\ell < k$, and $\gamma_0 \notin \widehat{\mathcal{D}}^k_{w\mathtt{R}v}$, otherwise.*

Proof. First observe that, if $\ell < k$, by Items 3 and 6 of Lemma 4.1, position γ_z belongs to both $\mathcal{D}^k_w$ and $\widehat{\mathcal{D}}^k_w$, since $0 < z < 2(k-|w|)$. Let us consider Item 1 of the current lemma first and show that the every position γ_z belongs to all the descendants of $\mathcal{D}^k_w$ in its left subtree. The proof proceeds by induction on the length of the sequence v and recall that, due to Item 2 (*resp.*, 5) of Lemma 4.1, the position with maximal priority in $\mathcal{D}^k_w$ (*resp.*, in $\widehat{\mathcal{D}}^k_w$) is α_z (*resp.*, α_{z+1}). Assume for the base case that $|v| = 0$. The thesis becomes $\gamma_z \in \mathcal{D}^k_{w\mathtt{L}}$ and $\gamma_z \in \widehat{\mathcal{D}}^k_{w\mathtt{L}}$. By Item 1 of Definition 4.4, $\widehat{\mathcal{D}}^k_{w\mathtt{L}}$ is defined as $\mathcal{D}^k_w \setminus \mathrm{A}$, where $\mathrm{A} = \mathsf{atr}^1_{\mathcal{D}^k_w}(\{\alpha_{z+1}\})$. By Definition 4.2 of core extension (Items 1, 3, and 4), a move entering α_{z+1} may only come from β_{z+2}, if it is present in the subgame, which is always the case unless $w = \varepsilon$, or from γ_z, whose owner is the opponent player 0 and cannot be attracted. Hence, $\mathrm{A} = \{\alpha_{z+1}, \beta_{z+2}\}$, if $w \neq \varepsilon$, and $\mathrm{A} = \{\alpha_{z+1}\}$, otherwise. Similarly, by Item 3, $\mathcal{D}^k_{w\mathtt{L}}$ is defined as $\widehat{\mathcal{D}}^k_{w\mathtt{L}} \setminus \mathrm{A}$, where $\mathrm{A} = \mathsf{atr}^{\mathsf{o}}_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}(\{\alpha_z\})$. For the same observations as in the previous case, we have that $\mathrm{A} = \{\alpha_z, \beta_{z+1}\}$. Hence, no position γ_i is removed

from either games and the thesis immediately follows. Assume now $|v| > 0$, let $v \triangleq v' \cdot x$, with $x \in \{\mathtt{L}, \mathtt{R}\}$, and set $\overline{w} \triangleq w\mathtt{L}v'$. By the inductive hypothesis, $\gamma_z \in \widehat{\mathcal{D}}^k_{\overline{w}}$ and $\gamma_z \in \mathcal{D}^k_{\overline{w}}$. We have two cases, depending on whether $x = \mathtt{L}$ or $x = \mathtt{R}$. Let $r \triangleq 2(k - |\overline{w}|)$. If $x = \mathtt{L}$, according to Item 1 of Definition 4.4, $\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}$ is obtained from $\mathcal{D}^k_{\overline{w}}$ by removing $\mathrm{A} = \mathsf{atr}^1_{\mathcal{D}^k_{\overline{w}}}(\{\alpha_r\}) = \{\alpha_r, \beta_{r+1}\}$. Similarly, by Item 3 of Definition 4.4, $\mathcal{D}^k_{\overline{w}\mathtt{L}}$ is defined as $\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}} \setminus \mathrm{A}$, where $\mathrm{A} = \mathsf{atr}^0_{\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}}(\{\alpha_{r-1}\})$, by observing that $|\overline{w}\mathtt{L}| = |\overline{w}| + 1$ and, therefore, $2(k - |\overline{w}\mathtt{L}|) + 1 = r - 1$. In both cases the thesis follows immediately. Let us now consider the case with $x = \mathtt{R}$. According to Item 2 of Definition 4.4, $\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{R}}$ is obtained by removing the set $\mathrm{A} = \mathsf{atr}^0_{\mathcal{D}^k_{\overline{w}}}(\mathrm{Wn}^0_{\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}})$ from $\mathcal{D}^k_{\overline{w}}$. Position γ_z has odd index and cannot belong to $\mathrm{Wn}^0_{\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}}$, which, by Item 7 of Lemma 4.1, only contains, among the positions from the core, those β_i and γ_i, with $i \geq 2(k - |w|) > z$ and even. Since, $\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}$ is a subgame of a core extension, it holds that γ_z is owned by player 0 and can only have a move leading to α_{z+1}, which is not in the subgame, or to a position outside the core and owned by player 1. As a consequence, it cannot end up in $\mathsf{atr}^0_{\mathcal{D}^k_{\overline{w}}}(\mathrm{Wn}^0_{\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}})$ and the thesis holds. Finally, recall that $\mathcal{D}^k_{\overline{w}\mathtt{R}} = \widehat{\mathcal{D}}^k_{\overline{w}\mathtt{R}} \setminus \mathrm{A}$, where $\mathrm{A} = \mathsf{atr}^0_{\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{R}}}(\{\alpha_{\hat{r}}\})$, with $\hat{r} \triangleq 2(k - |\overline{w}\mathtt{R}|) + 1$. In the considered subgame, $\alpha_{\hat{r}}$ has incoming moves only from $\beta_{\hat{r}+1}$ and $\gamma_{\hat{r}-1}$. However, $\hat{r} - 1 = 2(k - |\overline{w}\mathtt{R}|) < 2(k - |w|) - 1 = z$. Moreover, $\hat{r} + 1$ is an even index, and thus $\beta_{\hat{r}+1}$ is not contained in $\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{R}}$, being in $\mathrm{Wn}^0_{\widehat{\mathcal{D}}^k_{\overline{w}\mathtt{L}}}$ as shown above. As a consequence, $\mathrm{A} = \{\alpha_{\hat{r}}\}$ and the thesis follows. In addition, when $|w| = k$, Item 3 of Lemma 4.1 tells us that $\gamma_0 \in \mathcal{D}^k_w$. If $\ell = k$, instead, we have that γ_0 belongs to $\widehat{\mathcal{D}}^k_{w\mathtt{L}}$, due to Item 6 of Lemma 4.1. This ends the proof of Item 1 of the lemma. As to Item 2 of the lemma, first observe that, if $|w| < k$, then $\gamma_z \notin \widehat{\mathcal{D}}^k_{w\mathtt{R}}$. Indeed, position $\gamma_z \in \mathsf{atr}^0_{\mathcal{D}^k_w}(\mathrm{Wn}^0_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}})$, as shown above. Since this set is removed from $\mathcal{D}^k_w$ to obtain $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$, the thesis holds for $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$. Moreover, every descendant of $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$ in the subgame tree is obtained only by removing positions. As a consequence, none of them can contain position γ_z. In case $|w| = k$, instead, it suffices to observe that, according to Item 7 of Lemma 4.1, $\gamma_0 \in \mathrm{Wn}^0_{\widehat{\mathcal{D}}^k_{w\mathtt{L}}}$, hence it cannot be contained in $\widehat{\mathcal{D}}^k_{w\mathtt{R}}$. □

The main result asserting the exponential size of the induced subtrees of any worst-case family is given by the next lemma. This follows by observing that the number of nodes in the induced tree is exponential in k and by showing that the subgames associated with any two nodes in the tree $\mathbb{G}^k$ are indeed different.

Lemma 4.4. $\left|\mathbb{G}^k\right| = 3(2^{k+1} - 1)$, *for any* $k \in \mathbb{N}_+$.

Proof. To prove that the size of $\mathbb{G}^k$ is as stated, we first need to show that all the elements contained in the subgame trees are different, namely that, for each $w \neq w'$, the subgames $\mathcal{D}^k_w$, $\mathcal{D}^k_{w'}$, $\widehat{\mathcal{D}}^k_w$, and $\widehat{\mathcal{D}}^k_{w'}$ are pairwise different. Let us start by showing that $\mathcal{D}^k_w \neq \widehat{\mathcal{D}}^k_w$, for each $w \neq \varepsilon$. By Item 5 of Lemma 4.1, position $\alpha_{2(k-|w|)+1} \in \widehat{\mathcal{D}}^k_w$ and, by Item 3 of Definition 4.4, this position is removed from $\widehat{\mathcal{D}}^k_w$ to obtain $\mathcal{D}^k_w$. Hence, those two subgames cannot be equal. Let us consider now two subgames, each associated with one of the sequences w and w'. There are two possible cases: either *(i)* w is a strict prefix of w', *i.e.*, the one subgame is a descendant of the other in the subgame tree, or *(ii)* w and w' share a common longest prefix $\overline{w}$ that is different from both, *i.e.*, the two subgames lie in two distinct subtrees of the subgame associated with $\overline{w}$. In case *(i)* we have that $w' = wv$, for some $v \neq \varepsilon$. An easy induction on the length of v can prove that if $\mathcal{D} \in \{\mathcal{D}^k_w, \widehat{\mathcal{D}}^k_w\}$ and each $\mathcal{D}' \in \{\mathcal{D}^k_{w'}, \widehat{\mathcal{D}}^k_{w'}\}$ are the subgames associated with w and w', respectively, then the second is a strict subgame of the first, *i.e.*, $\mathcal{D}' \subset \mathcal{D}$. Indeed, Definition 4.4 together with Items 2, 5, and 7 of Lemma 4.1 ensure that, at each step downward along a path in the tree starting from $\mathcal{D}$, whether we proceed on the left or the right branch, at least one position is always removed from the current subgame. In case *(ii)*, instead, Item 1 of Lemma 4.3 tells us that there is at least one position, γ_z in the lemma, contained in all the subgames of the left subtree, while Item 2 states that the same position

is not contained in any subgame of the right subtree. Therefore, each of the two subgames associated with w must be different from either of the two subgames associated with w'.

Finally, to prove the statement of the lemma, it suffices to observe that the number of sequences of length at most k over the alphabet $\{\mathtt{L},\mathtt{R}\}$ are precisely $2^{k+1}-1$ and with each such sequence w a subgame $\mathfrak{D}^k_w$ is associated. As a consequence, the set $\{\mathfrak{D}^k_w\}_{w\in\{\mathtt{L},\mathtt{R}\}^{\leq k}}$ contains $2^{k+1}-1$ different elements. Moreover, each such subgame has two children in $\{\widehat{\mathfrak{D}}^k_w\}^{w\neq\varepsilon}_{w\in\{\mathtt{L},\mathtt{R}\}^{\leq k+1}}$. We can, then, conclude that the size of $\mathbb{G}^k$ is precisely $3(2^{k+1}-1)$. □

As a consequence of Lemma 4.4, we can obtain a stronger lower bound on the execution time of the Recursive algorithm. Indeed, the result holds regardless of whether the algorithm is coupled with a memoization technique.

Theorem 4.1 (Exponential Worst Case)**.** *The number of distinct recursive calls executed by the Recursive algorithm, with or without memoization, on a game with n positions is* $\Omega\big(2^{\frac{n}{6}}\big)$ *in the worst case.*

Proof. To prove the theorem, it suffices to consider a game $\mathfrak{D}^k_{\mathtt{C}}$ belonging to the core family. Indeed, Lemma 4.2 states that the induced subgame tree of $\mathfrak{D}^k_{\mathtt{C}}$ is a subset of the recursion tree induced by the Recursive algorithm executed on that game. Therefore, according to Lemma 4.4, the algorithm performs at least $3(2^{k+1}-1)$ calls, each on a different subgame. By Definition 4.1, game $\mathfrak{D}^k_{\mathtt{C}}$ has $n=6k+3$ positions and, therefore, we have $k=\frac{n}{6}-\frac{1}{2}$. As a consequence, the number of recursive calls is bounded from below by $3(2^{\frac{n}{6}+\frac{1}{2}}-1)=\Omega\big(2^{\frac{n}{6}}\big)$. □

5 SCC-Decomposition Resilient Games

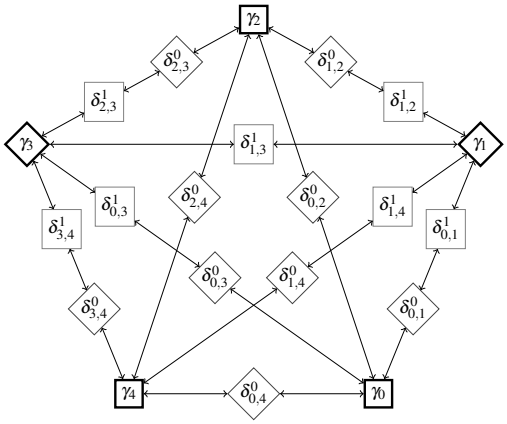

Figure 3: Clique of $\mathfrak{D}^2$.

The previous section provides a class of parametric families of parity games over which a dynamic-programming approach cannot help improving the asymptotic exponential behavior of the classic Recursive algorithm. However, it is not hard to observe that an SCC-decomposition of the underlying game graph, if applied by each recursive call as described in [20], would disrupt the recursive structure of the core family $\{\mathfrak{D}^k_{\mathtt{C}}\}^{\omega}_{k=1}$ and, consequently, break the exponential worst-case. This is due to the fact that the subgames $\mathfrak{D}^k_w$ in the induced subgame tree get decomposed into distinct SCCs, which can then be solved as independent subgames and memoized. In other words, the Recursive algorithm extended with memoization and SCC-decomposition can easily solve the core family. A concrete instance of this behavior can be observed by looking at the two games $\widehat{\mathfrak{D}}^1_{\mathtt{LL}}$ and $\widehat{\mathfrak{D}}^1_{\mathtt{RL}}$ of Figure 2. The first one is formed by three distinct components, one of which exactly corresponds to $\widehat{\mathfrak{D}}^1_{\mathtt{RL}}$. Therefore, a solution of these components immediately implies that the leaves in the induced subgame tree could not be considered distinct games *w.r.t.* to the behavior of the combined algorithm anymore. This behavior can, however, be prevented by introducing a suitable extension of the core family, which complies with the requirements of Definition 4.2. The basic idea is to connect all the pairs of positions γ_i and γ_j together in a clique-like fashion, by means of additional positions, denoted $\delta^{\wp}_{\{i,j\}}$ with $\wp\in\{0,1\}$, whose owners $\wp$ are chosen so as to preserve the exponential behavior on the underlying core family. With more detail, if $i\equiv_2 j$, there is a unique connecting position $\delta^{\wp}_{\{i,j\}}$ of parity $\wp\equiv_2 i$, the opposite of that of γ_i and γ_j. If, on the other hand, $i\not\equiv_2 j$, two mutually connected positions, $\{\delta^0_{\{i,j\}},\delta^1_{\{i,j\}}\}$, separate γ_i and γ_j. Figure 3

depicts the extension of the core game ∂^2_{C}, where, besides the positions γi, only the additional positions and their moves are shown. The complete formalization of the new family follows.

Definition 5.1 (SCC Family)**.** *The* SCC family $\{\partial^k_{\mathsf{S}}\}^{\omega}_{k=1}$*, where* $\partial^k_{\mathsf{S}} \triangleq \langle \mathscr{A}, \mathrm{Pr}, \mathsf{pr}\rangle$*,* $\mathscr{A} \triangleq \langle \mathrm{Ps}^0, \mathrm{Ps}^1, Mv\rangle$*,* $\mathrm{Pr} \triangleq [0,4k]$*, and* $\mathrm{Ps} \triangleq \partial^k_{\mathsf{C}} \cup \mathrm{P}$*, is defined, for any index* $k \geq 1$*, as follows:*

1. $\mathrm{P} \triangleq \{\delta^{\wp}_{\{i,j\}} : \wp \in \{0,1\} \wedge i,j \in [0,2k] \wedge i \neq j \wedge (i \equiv_2 j \rightarrow \wp \equiv_2 i)\}$*;*
2. $(\gamma_i, \delta^{\wp}_{\mathrm{I}}), (\delta^{\wp}_{\mathrm{I}}, \gamma_i) \in Mv$ iff $i \in \mathrm{I}$ *and* $i \equiv_2 \wp$*, for* $i \in [0,2k]$ *and* $\delta^{\wp}_{\mathrm{I}} \in \mathrm{P}$*;*
3. $(\delta^{\wp}_{\mathrm{I}}, \delta^{\overline{\wp}}_{\mathrm{I}}) \in Mv$ iff $i \not\equiv_2 j$*, for* $\delta^{\wp}_{\mathrm{I}} \in \mathrm{P}$ *and* $\mathrm{I} = \{i,j\}$*;*
4. $\delta^{\wp}_{\mathrm{I}} \in \mathrm{Ps}^{\wp}$ *and* $\mathsf{pr}(\delta^{\wp}_{\mathrm{I}}) \triangleq 0$*, for* $\delta^{\wp}_{\mathrm{I}} \in \mathrm{P}$*;*
5. $\partial^k_{\mathsf{S}} \setminus \mathrm{P} = \partial^k_{\mathsf{C}}$.

Intuitively, in Item 1, P denotes the set of additional positions of ∂^k_{S} *w.r.t.* to the core family game ∂^k_{C}, which is, indeed, a proper subgame, as stated in Item 5. Item 2, instead, formalizes the moves connecting the additional $\delta^{\wp}_{\mathrm{I}}$ positions with the γ_i of the core, while Item 3 describes the mutual connection between the δ positions that share the same doubleton of indexes I. Finally, Item 4 associates each $\delta^{\wp}_{\mathrm{I}}$ with its corresponding owner $\wp$ and priority 0. The following lemma proves that such a parity-game family is indeed a worst-case family.

Lemma 5.1. *The SCC family* $\{\partial^k_{\mathsf{S}}\}^{\omega}_{k=1}$ *is a worst-case family.*

Proof. To prove that the *SCC family* of Definition 5.1 is a worst-case family, we need to show that each game ∂^k_{S} is a core extension of ∂^k_{C}, *i.e.*, that it complies with the Definition 4.2. Items 1 and 5 of Definition 5.1 imply Item 1 of Definition 4.2, since the set P does not contain any position of the core and, in addition, ∂^k_{C} is a subgame of ∂^k_{S}, as all the positions and moves of the core are contained in ∂^k_{S}. Item 2 of Definition 4.2 follows from Item 4 of Definition 5.1. Indeed, by Definition 4.1, $\mathsf{pr}(\alpha_i) \triangleq 2k+i+1$, for $i \in [0,2k]$ and $k \geq 1$, while all the additional positions in P have priority 0. By Items 2 and 3 of Definition 5.1, there are no moves connecting positions δ_{I} with positions α_i or β_i, for any $i \in [0,2k]$, hence Item 3 of Definition 4.2 is satisfied. Finally, we need to show that whenever a γ_i has a move to a position v in P, then v does not have higher priority, belongs to the opponent of γ_i, and has a move back to γ_i (Item 4 of Definition 4.2). This property is enforced by Items 2 and 4 of Definition 5.1. Indeed, by the latter, position $\delta^{\wp}_{\mathrm{I}}$ is owned by player $\wp$. Moreover, by the former, γ_i, whose owner is $(i+1) \bmod 2$, can only have a move to $\delta^{\wp}_{\{i,j\}}$ if $\delta^{\wp}_{\{i,j\}}$ has a move back to γ_i and $\wp = i \bmod 2$. Hence, the two positions belong to opposite players. Since, in addition, all positions δ have priority 0, the requirement is satisfied. □

Due to the clique-like structure of the new family, it is not hard to see that every game in the induced subgame tree forms a single SCC. This guarantees that the intertwining of SCC-decomposition and memoization cannot prevent an exponential worst-case behavior of the Recursive algorithm on this family.

Lemma 5.2. *Each game in the induced subgame tree* $\mathbb{G}^k$ *of the SCC family* $\{\partial^k_{\mathsf{S}}\}^{\omega}_{k=1}$*, for an arbitrary index* $k \in \mathbb{N}_+$*, forms a single SCC.*

Proof. Let $\partial^k_w \in \mathbb{G}^k$ (*resp.*, $\widehat{\partial}^k_w \in \mathbb{G}^k$) be a game in the induced subgame tree. By induction on the structure of the string w, it is not hard to see that, for all indexes $i,j \in [0,2k]$ with $i \neq j$, it holds that $\gamma_i, \gamma_j \in \partial^k_w$ (*resp.*, $\gamma_i, \gamma_j \in \widehat{\partial}^k_w$) iff the positions $\delta^{\wp}_{\{i,j\}} \in \mathrm{P}$, with $\wp \in \{0,1\}$, belong to ∂^k_w (*resp.*, $\widehat{\partial}^k_w$), as well. For the base case $\partial^k_{\varepsilon} = \partial^k$, the thesis trivially follows from Definition 5.1. For the inductive case $\widehat{\partial}^k_{wx} = \partial^k_w \setminus \mathrm{A}$ (*resp.*, $\partial^k_{wx} = \widehat{\partial}^k_w \setminus \mathrm{A}$), let us assume, as inductive hypothesis, that the statement holds for ∂^k_w (*resp.*, $\widehat{\partial}^k_w$). By Definition 4.4, the set A is computed as the attractor to some set of positions B such that either

(i) $\gamma_i, \gamma_j, \delta^{\wp}_{\{i,j\}} \notin \mathrm{A}$ or *(ii)* $\delta^{\wp}_{\{i,j\}} \in \mathrm{A}$ and at least one between γ_i and γ_j belongs to A, for all $i, j \in [0, 2k]$. Case *(i)* arises when $\mathrm{B} = \{\alpha_{2(k-|w|)}\}$ (*resp.*, $\mathrm{B} = \{\alpha_{2(k-|w|)+1}\}$), while Case *(ii)* when $\mathrm{B} = \mathrm{Wn}_0(\widehat{\mathfrak{I}}^k_{w\mathrm{L}})$. Consequently, the required property on *wx* immediately follows from the inductive hypothesis on *w*, since, if a position $\delta^{\wp}_{\{i,j\}}$ is removed from the game, also one between γ_i and γ_j is removed as well and *vice versa*. Now, let $\mathfrak{I}$ be an arbitrary game in $\mathbb{G}^k$. Thanks to the topology of the games in the SCC family and to the property proved above, it easy to see that all positions in $\mathrm{X} = \{\beta_i, \gamma_i, \delta^{\wp}_{\mathrm{I}} \in \mathfrak{I}\}$ form a strongly connected subgame. Indeed, two positions β_i and γ_i are mutually reachable due to the two moves $(\beta_i, \gamma_i), (\gamma_i, \beta_i) \in Mv$. Moreover, two arbitrary positions γ_i and γ_j, with $i \neq j$, are mutually reachable via the positions $\delta^{\wp}_{\{i,j\}}$. Also, there are no isolated positions $\delta^{\wp}_{\{i,j\}}$. Finally, to prove that $\mathfrak{I}$ is indeed a single SCC, it remains just to show that the positions in $\mathrm{Y} = \{\alpha_i \in \mathfrak{I}\}$ can reach and can be reached by those in X. The first part is implied by Item 1 of Lemma 4.1, since every α_i has only a move to β_i, which needs to belong to $\mathfrak{I}$ in order for this to be a game. Now, due to the same observation, all positions α_i, but possibly the last one α_m with maximal index m in $\mathfrak{I}$, are reachable by β_{i+1}. Finally, α_m can be reached by γ_{m-1}, which necessarily belongs to $\mathfrak{I}$ due to Item 3 of the same lemma. □

Putting everything together, we obtain the following structural, although non asymptotic, strengthening of Theorem 4.1.

Theorem 5.1 (SCC-Decomposition Worst Case)**.** *The number of distinct recursive calls executed by the Recursive algorithm with SCC decomposition on a game with n positions is* $\Omega\left(2^{\sqrt{n/3}}\right)$ *in the worst case.*

Proof. Due to Lemma 5.1, the *SCC family* is an exponential worst-case family. As shown in the proof of Theorem 4.1, the Recursive algorithm performs at least $3(2^{k+1}-1)$ calls to solve a game of $\{\mathfrak{I}^k\}^{\omega}_{k=1}$, and therefore, of $\{\mathfrak{I}^k_{\mathrm{S}}\}^{\omega}_{k=1}$. As consequence of Lemma 5.2, the number of calls cannot be affected by an SCC-decomposition technique, since there is an exponential number of subgames, the ones in the induced subgame tree of $\mathfrak{I}^k_{\mathrm{S}}$, each of which forms a single SCC. Moreover, a core game $\mathfrak{I}^k_{\mathrm{C}}$ has $6k+3$ positions, while the number of additional positions P in the extension $\mathfrak{I}^k_{\mathrm{C}}$ is given by $3k^2+2k$, which follows from Definition 5.1. Indeed, for every pair of positions γ_i, γ_j, with $i, j \in [0, 2k]$ and $i \neq j$, there is a single position $\delta^{\wp}_{\{i,j\}}$, if $i \equiv_2 j$, and two such positions, otherwise. Hence, $\mathfrak{I}^k_{\mathrm{S}}$ has $n \triangleq 3k^2+8k+3$ positions, from which we obtain that $k = \frac{\sqrt{3n+7}-4}{3}$. As a consequence, the number of recursive calls is bounded from below by $3(2^{\frac{\sqrt{3n+7}-1}{3}}-1) = \Omega\left(2^{\sqrt{n/3}}\right)$. □

6 Dominion-Decomposition Resilient Games

A deeper analysis of the SCC family reveals that the size of the smallest dominion for player 0 in Game $\mathfrak{I}^k_{\mathrm{S}}$ (there are no dominions for player 1, being the game completely won by its opponent) is of size $2(k+1)$. This observation, together with the fact that the game has $3k^2+8k+3$ positions, immediately implies that the proposal of [26, 27] of a brute force search for dominions of size at most $\left\lceil\sqrt{3k^2+8k+3}\right\rceil < 2(k+1)$ cannot help improving the solution process on these games. We can prove an even stronger result, since this kind of search cannot reduce the running time in any of the subgames of the induced tree. The reason is that the smallest dominion in each such subgame that contains at least a position in the core has size linear *w.r.t.* k, as reported by the following lemma.

Lemma 6.1. *For all* $k \in \mathbb{N}_+$*, let* $\mathfrak{I}$ *be a subgame in the induced subgame tree* $\mathbb{G}^k$ *of* $\mathfrak{I}^k_{\mathrm{S}}$*, and* $\mathrm{D} \subseteq \mathfrak{I}$ *the smallest dominion such that* $\mathrm{D} \cap \mathfrak{I}^k_{\mathrm{C}} \neq \emptyset$*. Then,* $|\mathrm{D}| \geq k+1$.

Proof. We start by proving that any such dominion D must necessarily contain at least a position γ_i, from some $i \in \mathbb{N}_+$. Assume, by contradiction, that it does not. Then, $\mathrm{D} \subseteq \{\alpha_i, \beta_i \in \eth : i \in [0,2k]\} \cup \mathrm{P}$. We can prove that D is not a game. By assumption, D must contain at least one position α_i or β_i, otherwise $\mathrm{D} \cap \eth^k_{\mathrm{C}} = \emptyset$. Let j be the smallest index such that $\alpha_j \in \mathrm{D}$ or $\beta_j \in \mathrm{D}$. Then, at least one of those two positions has only moves leading outside D. Hence D is not a game and, *a fortiori*, cannot be a dominion.

Therefore, D must contain at least a position γ_i. By Definition 5.1, γ_i is connected in $\eth^k_{\mathrm{S}}$ to precisely $2k$ positions $\delta^{\wp}_{\{i,j\}}$, where $j \in [0,2k]$, $j \neq i$, and whose owner $\wp \equiv_2 i$ is the adversary of the owner of γ_i. Let us consider the k indexes $j \in [0,2k]$ such that $i \not\equiv_2 j$. For each such j, we have two cases, depending on whether γ_j belongs to $\eth$ or not. If it does, then so does $\delta^{\wp}_{\{i,j\}}$. If it does not, then it must have been removed by some application of Item 2 of Definition 4.4. If the involved attractor *w.r.t.* player 0 does not attract $\delta^{\overline{\wp}}_{\{i,j\}}$, then it cannot attract $\delta^{\wp}_{\{i,j\}}$ either, as it has no moves to γ_j. If, on the other hand, $\delta^{\overline{\wp}}_{\{i,j\}}$ gets attracted, it must belong to player 0. As a consequence, $\delta^{\wp}_{\{i,j\}}$ belongs to player 1 and is not attracted. In either case, we conclude hat $\delta^{\wp}_{\{i,j\}}$ cannot be removed and, therefore, is still contained in $\eth$. Since, in addition, all the k positions $\delta^{\wp}_{\{i,j\}}$, with $j \in [0,2k]$ and $j \not\equiv_2 i$, are mutually connected to γ_i and their owner is the opponent of the one of γ_i, they must be contained in D as well. Hence, D contains at least $k+1$ positions. □

The above observation allows us to obtain an exponential lower bound for the Recursive algorithm combined with memoization, SCC decomposition, and dominion decomposition techniques. Indeed, the brute-force procedure employed by the Dominion Decomposition algorithm of [26] needs at least time $\Omega\left(2^{k+1}\right)$ to find a dominion of size $k+1$. For the sake of space and clarity of exposition, we postpone the formal treatment of the Big Step procedure to the extended version of this work, reporting here an informal description only. Intuitively, its exponential behavior on the SCC family follows from the following observations: *(i)* the algorithm only looks for dominions of size $d = \sqrt[3]{pn^2}$, where p and n are the numbers of priorities and positions, and *(ii)* the update of the measure functions used by the procedure requires an exponential number of steps in d before reaching a fixpoint. As a consequence, none of the dominion decomposition approaches, combined with memoization and SCC decomposition, can efficiently solve the SCC family.

Corollary 6.1 (Exponential Dominion-Decomposition Worst Case)**.** *The solution time of the Recursive algorithm with memoization, SCC decomposition, and dominion decomposition on a game with n positions is* $\Omega\left(2^{\Theta(\sqrt{n})}\right)$ *in the worst case.*

References

[1] R. Alur, T.A. Henzinger & O. Kupferman (2002): *Alternating-Time Temporal Logic.* *JACM* 49(5), pp. 672–713, doi:10.1145/585265.585270.

[2] K. Apt & E. Grädel (2011): *Lectures in Game Theory for Computer Scientists.* Cambridge University Press, doi:10.1017/CBO9780511973468.

[3] L. Babai (2016): *Graph Isomorphism in Quasipolynomial Time [Extended Abstract].* In: *STOC'16*, ACM, pp. 684–697, doi:10.1145/2897518.2897542.

[4] M. Benerecetti, D. Dell'Erba & F. Mogavero: *Solving Parity Games via Priority Promotion.* Under submission.

[5] M. Benerecetti, D. Dell'Erba & F. Mogavero (2016): *A Delayed Promotion Policy for Parity Games.* In: *GANDALF16*, EPTCS 226, pp. 30–45, doi:10.4204/EPTCS.226.3.

[6] M. Benerecetti, D. Dell'Erba & F. Mogavero (2016): *Improving Priority Promotion for Parity Games.* In: *HVC'16*, LNCS 10028, Springer, pp. 1–17, doi:10.1007/978-3-319-49052-6_8.

[7] M. Benerecetti, D. Dell'Erba & F. Mogavero (2016): *Solving Parity Games via Priority Promotion.* In: *CAV'16*, LNCS 9780 (Part II), Springer, pp. 270–290, doi:10.1007/978-3-319-41540-6_15.

[8] M. Benerecetti, F. Mogavero & A. Murano (2013): *Substructure Temporal Logic.* In: *LICS'13*, IEEECS, pp. 368–377, doi:10.1109/LICS.2013.43.

[9] M. Benerecetti, F. Mogavero & A. Murano (2015): *Reasoning About Substructures and Games. TOCL* 16(3), pp. 25:1–46, doi:10.1109/LICS.2013.43.

[10] D. Berwanger & E. Grädel (2004): *Fixed-Point Logics and Solitaire Games. TCS* 37(6), pp. 675–694, doi:10.1007/s00224-004-1147-5.

[11] C.S. Calude, S. Jain, B. Khoussainov, W. Li & F. Stephan (2017): *Deciding Parity Games in Quasipolynomial Time.* In: *STOC'17*, ACM, pp. 252–263, doi:10.1145/3055399.3055409.

[12] K. Chatterjee, T.A. Henzinger & N. Piterman (2010): *Strategy Logic. IC* 208(6), pp. 677–693, doi:10.1016/j.ic.2009.07.004.

[13] A. Condon (1992): *The Complexity of Stochastic Games. IC* 96(2), pp. 203–224, doi:10.4230/LIPIcs.FSTTCS.2010.505.

[14] A. Ehrenfeucht & J. Mycielski (1979): *Positional Strategies for Mean Payoff Games. IJGT* 8(2), doi:10.1007/BF01768705.

[15] E.A. Emerson & C.S. Jutla (1991): *Tree Automata, muCalculus, and Determinacy.* In: *FOCS'91*, IEEECS, pp. 368–377, doi:10.1109/SFCS.1988.21949.

[16] E.A. Emerson, C.S. Jutla & A.P. Sistla (2001): *On Model Checking for the muCalculus and its Fragments. TCS* 258(1-2), pp. 491–522, doi:10.1016/S0304-3975(00)00034-7.

[17] E.A. Emerson & C.-L. Lei (1986): *Temporal Reasoning Under Generalized Fairness Constraints.* In: *STACS'86*, LNCS 210, Springer, pp. 267–278, doi:10.1007/3-540-16078-7_62.

[18] J. Fearnley, S. Jain, S. Schewe, F. Stephan & D. Wojtczak (2017): *An Ordered Approach to Solving Parity Games in Quasi Polynomial Time and Quasi Linear Space.* In: *SPIN'17*, ACM, pp. 112–121, doi:10.1145/3092282.3092286.

[19] O. Friedmann (2011): *Recursive Algorithm for Parity Games Requires Exponential Time. RAIROTIA* 45(4), pp. 449–457, doi:10.1051/ita/2011124.

[20] O. Friedmann & M. Lange (2009): *Solving Parity Games in Practice.* In: *ATVA'09*, LNCS 5799, Springer, pp. 182–196, doi:10.1007/978-3-642-04761-9_15.

[21] E. Grädel, W. Thomas & T. Wilke (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research.* LNCS 2500, Springer, doi:10.1007/3-540-36387-4.

[22] V.A. Gurvich, A.V. Karzanov & L.G. Khachivan (1990): *Cyclic Games and an Algorithm to Find Minimax Cycle Means in Directed Graphs. USSRCMMP* 28(5), pp. 85–91, doi:10.1016/0041-5553(88)90012-2.

[23] M. Jurdziński (1998): *Deciding the Winner in Parity Games is in UP ∩ co-UP. IPL* 68(3), pp. 119–124, doi:10.1016/S0020-0190(98)00150-1.

[24] M. Jurdziński (2000): *Small Progress Measures for Solving Parity Games.* In: *STACS'00*, LNCS 1770, Springer, pp. 290–301, doi:10.1007/3-540-46541-3_24.

[25] M. Jurdziński & R. Lazic (2017): *Succinct Progress Measures for Solving Parity Games.* In: *LICS'17*, ACM, pp. 1–9, doi:10.1109/LICS.2017.8005092.

[26] M. Jurdziński, M. Paterson & U. Zwick (2006): *A Deterministic Subexponential Algorithm for Solving Parity Games.* In: *SODA'06*, SIAM, pp. 117–123, doi:10.1145/1109557.1109571.

[27] M. Jurdziński, M. Paterson & U. Zwick (2008): *A Deterministic Subexponential Algorithm for Solving Parity Games. SJM* 38(4), pp. 1519–1532, doi:10.1137/070686652.

[28] N. Klarlund & D. Kozen (1991): *Rabin Measures and Their Applications to Fairness and Automata Theory*. In: *LICS'91*, IEEECS, pp. 256–265, doi:10.1109/LICS.1991.151650.

[29] O. Kupferman & M.Y. Vardi (1998): *Weak Alternating Automata and Tree Automata Emptiness.* In: *STOC'98*, ACM, pp. 224–233, doi:10.1145/276698.276748.

[30] A.D. Martin (1975): *Borel Determinacy. AM* 102(2), pp. 363–371.

[31] A.D. Martin (1985): *A Purely Inductive Proof of Borel Determinacy.* In: *SPM'82*, Recursion Theory., AMS and ASL, pp. 303–308.

[32] R. McNaughton (1993): *Infinite Games Played on Finite Graphs. APAL* 65, pp. 149–184, doi:10.1016/0168-0072(93)90036-D.

[33] F. Mogavero, A. Murano, G. Perelli & M.Y. Vardi (2012): *What Makes ATL* Decidable? A Decidable Fragment of Strategy Logic.* In: *CONCUR'12*, LNCS 7454, Springer, pp. 193–208, doi:10.1007/978-3-642-32940-1_15.

[34] F. Mogavero, A. Murano, G. Perelli & M.Y. Vardi (2014): *Reasoning About Strategies: On the Model-Checking Problem. TOCL* 15(4), pp. 34:1–42, doi:10.1145/2631917.

[35] F. Mogavero, A. Murano, G. Perelli & M.Y. Vardi (2017): *Reasoning About Strategies: On the Satisfiability Problem. LMCS* 13(1:9), pp. 1–37, doi:10.23638/LMCS-13(1:9)2017.

[36] F. Mogavero, A. Murano & M.Y. Vardi (2010): *Reasoning About Strategies.* In: *FSTTCS'10*, LIPIcs 8, Leibniz-Zentrum fuer Informatik, pp. 133–144, doi:10.4230/LIPIcs.FSTTCS.2010.133.

[37] A.W. Mostowski (1984): *Regular Expressions for Infinite Trees and a Standard Form of Automata.* In: *SCT'84*, LNCS 208, Springer, pp. 157–168, doi:10.1007/3-540-16066-3_15.

[38] A.W. Mostowski (1991): *Games with Forbidden Positions.* Technical Report, University of Gdańsk, Gdańsk, Poland.

[39] D. Neuen & P. Schweitzer (2017): *An Exponential Lower Bound for Individualization-Refinement Algorithms for Graph Isomorphism.* Technical Report, arXiv.

[40] D. Neuen & P. Schweitzer (2017): *Benchmark Graphs for Practical Graph Isomorphism.* Technical Report, arXiv.

[41] S. Schewe (2007): *Solving Parity Games in Big Steps.* In: *FSTTCS'07*, LNCS 4855, Springer, pp. 449–460, doi:10.1007/978-3-540-77050-3_37.

[42] S. Schewe (2008): *ATL* Satisfiability is 2ExpTime-Complete.* In: *ICALP'08*, LNCS 5126, Springer, pp. 373–385, doi:10.1007/978-3-540-70583-3_31.

[43] S. Schewe & B. Finkbeiner (2006): *Satisfiability and Finite Model Property for the Alternating-Time muCalculus.* In: *CSL'06*, LNCS 6247, Springer, pp. 591–605, doi:10.1007/11874683_39.

[44] J. Vöge & M. Jurdziński (2000): *A Discrete Strategy Improvement Algorithm for Solving Parity Games.* In: *CAV'00*, LNCS 1855, Springer, pp. 202–215, doi:10.1007/10722167_18.

[45] T. Wilke (2001): *Alternating Tree Automata, Parity Games, and Modal muCalculus. BBMS* 8(2), pp. 359–391.

[46] W. Zielonka (1998): *Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. TCS* 200(1-2), pp. 135–183, doi:10.1016/S0304-3975(98)00009-7.

[47] U. Zwick & M. Paterson (1996): *The Complexity of Mean Payoff Games on Graphs. TCS* 158(1-2), pp. 343–359, doi:10.1016/0304-3975(95)00188-3.