



Universal Algorithms for Parity Games and Nested Fixpoints

Marcin Jurdziński, Rémi Morvan, K. S. Thejaswini

► To cite this version:

Marcin Jurdziński, Rémi Morvan, K. S. Thejaswini. Universal Algorithms for Parity Games and Nested Fixpoints. Lecture Notes in Computer Science, Springer, In press. hal-03762990

HAL Id: hal-03762990

<https://hal.archives-ouvertes.fr/hal-03762990>

Submitted on 29 Aug 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSAL ALGORITHMS FOR PARITY GAMES AND NESTED FIXPOINTS*

Marcin Jurdziński¹, Rémi Morvan^{2,3}, and K. S. Thejaswini¹

¹DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF WARWICK

²ÉCOLE NORMALE SUPÉRIEURE PARIS-SACLAY

³LABRI, UNIV. BORDEAUX, CNRS & BORDEAUX INP

August 2022

ABSTRACT

An [attractor decomposition](#) meta-algorithm for solving [parity games](#) is given that generalises the classic McNaughton-Zielonka algorithm and its recent [quasi-polynomial](#) variants due to Parys (2019), and to Lehtinen, Schewe, and Wojtczak (2019). The central concepts studied and exploited are [attractor decompositions](#) of [dominia](#) in [parity games](#) and the ordered trees that describe the inductive structure of [attractor decompositions](#).

The [universal algorithm](#) yields [McNaughton-Zielonka](#), [Parys](#), and [Lehtinen-Schewe-Wojtczak](#) algorithms as special cases when suitable [universal trees](#) are given to it as inputs. The main technical results provide a unified proof of correctness and structural insights into those algorithms.

Suitably adapting the [universal algorithm](#) for [parity games](#) to [fixpoint games](#) gives a [quasi-polynomial](#) time algorithm to compute [nested fixpoints](#) over finite complete lattices.

The [universal algorithms](#) for [parity games](#) and [nested fixpoints](#) can be implemented symbolically. It is shown how this can be done with $O(\lg d)$ [symbolic space](#) complexity, improving the $O(d \lg n)$ [symbolic space](#) complexity achieved by Chatterjee, Dvořák, Henzinger, and Svozil (2018) for [parity games](#), where n is the number of vertices and d is the number of distinct [priorities](#) in a [parity game](#).

Keywords: parity games, universal trees, attractor decompositions, quasi-polynomial, fixpoint equations, symbolic algorithms.

Acknowledgements. The first and the third author had been supported by the EPSRC grant EP/P020992/1 (Solving Parity Games in Theory and Practice). The idea of the design of the universal algorithm has been discovered independently and later by Nathanaël Fijalkow; we thank him for sharing his conjectures with us and exchanging ideas about adaptive tree-pruning rules. We also thank our anonymous reviewers and Alexander Kozachinskiy for helpful comments on earlier drafts of the paper.

This document contains [internal hyperlinks](#), and is best read on an electronic device.

*First version: January 2020. Full version of a paper accepted in Lecture Notes in Computer Science, volume 13660. Emails: marcin.jurdzinski@warwick.ac.uk, remi.morvan@ens-paris-saclay.fr and thejaswini.raghavan.1@warwick.ac.uk.

1. CONTEXT

1.1. PARITY GAMES AND THEIR SIGNIFICANCE

Parity games play a fundamental role in automata theory, logic, and their applications to verification [EJ91], program analysis [BKMMMP19, HS21], and synthesis [GTW02, LMS20]. In particular, parity games are very intimately linked to the problems of emptiness and complementation of non-deterministic automata on trees [EJ91, Zie98], model checking and satisfiability checking of fixpoint logics [EJ91, EJS93, BW18], fair simulation relations [EWS05] or evaluation of nested fixpoint expressions [HSC16, BKMMMP19, HS21]. It is a long-standing open problem whether parity games can be solved in polynomial time [EJS93].

The impact of parity games goes well beyond their home turf of automata theory, logic, and formal methods. For example, an answer [Fri09] of a question posed originally for parity games [VJ00] has strongly inspired major breakthroughs on the computational complexity of fundamental algorithms in stochastic planning [Fea10] and linear optimization [Fri11b, FHZ11], and parity games provide the foundation for the theory of nested fixpoint expressions used in program analysis [BKMMMP19, HS21] and coalgebraic model checking [HSC16].

1.2. RELATED WORK

The major breakthrough in the study of algorithms for solving parity games occurred in 2017 when Calude, Jain, Khoussainov, Li, and Stephan [CJK⁺17] have discovered the first quasi-polynomial algorithm. Three other—and seemingly distinctly different—techniques for solving parity games in quasi-polynomial time have been proposed in quick succession soon after: by Jurdziński and Lazić [JL17], Lehtinen [Leh18], and Lehtinen, Parys, Schewe, and Wojtczak [LPSW22]. We would like to remark that [LPSW22] is journal paper—describing two quasi-polynomial time algorithms—combining a conference paper of Parys [Par19] and a preprint by Lehtinen, Schewe, and Wojtczak [LSW19]. To distinguish between the two algorithms, we refer to these versions as the algorithms by *Parys* and by *Lehtinen-Schewe-Wojtczak*, respectively.

Czerwiński, Daviaud, Fijalkow, Jurdziński, Lazić, and Parys [CDF⁺19] have also uncovered an underlying combinatorial structure of universal trees as provably underlying the techniques of Calude et al., of Jurdziński and Lazić, and of Lehtinen. Czerwiński et al. have also established a quasi-polynomial lower bound for the size of smallest universal trees, providing evidence that the techniques developed in those three papers may be insufficient for leading to further improvements in the complexity of solving parity games. The work of Lehtinen, Parys, Schewe, and Wojtczak [LPSW22], who noted that the tree of recursive calls of their algorithms is universal, has not been obviously subject to the quasi-polynomial barrier of Czerwiński et al. [CDF⁺19], making it a focus of current activity. Their algorithms are obtained by modifying the classic McNaughton-Zielonka algorithm [McN93, Zie98], which has exponential running time in the worst case [Fri11a], but consistently outperforms most other algorithms in practice [vD18].

Using these universal trees as a crucial structure, there have also been further work to solve nested fixpoint expressions [HS21, ANP21] in quasi-polynomial time.

1.3. OUR CONTRIBUTIONS

In this work we provide a meta-algorithm—the [universal attractor decomposition algorithm](#)—that generalizes [McNaughton-Zielonka](#), [Parys’s](#), and [Lehtinen-Schewe-Wojtczak](#) algorithms. There are multiple benefits of considering the [universal algorithm](#).

Firstly, in contrast to [Parys’s](#) and [Lehtinen-Schewe-Wojtczak](#) algorithms, the [universal algorithm](#) has a very simple and transparent structure that minimally departs from the classic [McNaughton-Zielonka algorithm](#). Secondly, we observe that [Lehtinen-Schewe-Wojtczak algorithm](#), as well as non-adaptive versions (see Sections 3.2 and 4.4) of [McNaughton-Zielonka](#) and [Parys’s](#) algorithms, all arise from the [universal algorithm](#) by using specific classes of [universal trees](#), strongly linking the theory of [universal trees](#) to the only class of quasi-polynomial algorithms that had no established formal relationship to [universal trees](#) so far. Moreover, since our algorithm can be modified to use any trees, they can also run on several classes of [universal trees](#) like the Strahler universal trees introduced in the work of Daviaud, Jurdziński and Thejaswini [DJT20].

Thirdly, we further develop the theory of [dominia](#) and their [attractor decompositions in parity games](#), initiated by Daviaud, Jurdziński, and Lazić [DJL18] and by Daviaud, Jurdziński, and Lehtinen [DJL19], and we prove two new structural theorems (the [embeddable decomposition theorem](#) and the [dominion separation theorem](#)) about ordered trees of [attractor decompositions](#).

Fourthly, we use the structural theorems to provide a unified proof of correctness of various [McNaughton-Zielonka](#)-style algorithms, identifying very precise structural conditions on the trees of recursive calls of the [universal algorithm](#) that result in it correctly identifying the largest [dominia](#).

Fifthly, we identify a structure of nested [fixpoint games](#), the [parity games](#) that arise naturally while solving fixpoint expressions which help us solve them in quasi-polynomial time using a modification of our [universal algorithm](#).

Finally, we observe that thanks to its simplicity, the [universal algorithm](#) is particularly well-suited for solving [parity games](#) as well as [nested fixpoint equations](#) efficiently in a symbolic model of computation, when large sizes of input graphs prevent storing them explicitly in memory. Indeed, we argue that already a routine implementation of the [universal algorithm](#) for [parity games](#) improves the state-of-the-art [symbolic space complexity](#) of solving [parity games](#) in quasi-polynomial time from $O(d \lg n)$ to $O(d)$, but we also show that a more sophisticated symbolic data structure allows to further reduce the [symbolic space](#) of the [universal algorithm](#) to $O(\lg d)$.

2. DOMINIA AND DECOMPOSITIONS

2.1. STRATEGIES, TRAPS, AND DOMINIA

A [parity game](#) \mathcal{G} consists of a finite directed graph (V, E) together with a partition $(V_{\text{Even}}, V_{\text{Odd}})$ of the set of vertices V , and a function $\pi : V \rightarrow \{0, 1, \dots, d\}$ that labels every vertex $v \in V$ with a non-negative integer $\pi(v)$ called its [priority](#). We say that a cycle is *even* if the highest vertex [priority](#) on the cycle is even; otherwise the cycle is *odd*. We say that a [parity game](#) is (n, d) -*small* if it has at most n vertices and all vertex [priorities](#) are at most d .

For a set S of vertices, we write $\mathcal{G} \cap S$ for the substructure of \mathcal{G} whose graph is the

subgraph of (V, E) induced by the sets of vertices S . Sometimes, we also write $\mathcal{G} \setminus S$ to denote $\mathcal{G} \cap (V \setminus S)$. We assume throughout that every vertex has at least one outgoing edge, and we reserve the term *subgame* to substructures $\mathcal{G} \cap S$, such that every vertex in the subgraph of (V, E) induced by S has at least one outgoing edge. For a *subgame* $\mathcal{G}' = \mathcal{G} \cap S$, we sometimes write $V^{\mathcal{G}'}$ for the set of vertices S that the *subgame* \mathcal{G}' is induced by. When convenient and if the risk of confusion is contained, we may simply write \mathcal{G}' instead of $V^{\mathcal{G}'}$.

A (positional) Even *strategy* is a set $\sigma \subseteq E$ of edges such that:

- for every $v \in V_{\text{Even}}$, there is an edge $(v, u) \in \sigma$,
- for every $v \in V_{\text{Odd}}$, if $(v, u) \in E$ then $(v, u) \in \sigma$.

We sometimes call all the edges in such an Even *strategy* σ the *strategy edges*, and the definition of an Even *strategy* requires that every vertex in V_{Even} has an outgoing strategy edge, and every outgoing edge of a vertex in V_{Odd} is a strategy edge.

For a non-empty set of vertices T , we say that an Even *strategy* σ *traps Odd in* T if no strategy edge leaves T , that is, $w \in T$ and $(w, u) \in \sigma$ imply $u \in T$. We say that a set of vertices T is a *trap* for Odd if there is an Even *strategy* that traps Odd in T .

Observe that if T is a *trap* in a game \mathcal{G} then $\mathcal{G} \cap T$ is a *subgame* of \mathcal{G} . For brevity, we sometimes say that a *subgame* \mathcal{G}' is a *trap* if $\mathcal{G}' = \mathcal{G} \cap T$ and the set T is a *trap* in \mathcal{G} . Moreover, the following simple “*trap transitivity*” property holds: if T is a *trap* for Odd in game \mathcal{G} and T' is a *trap* for Odd in *subgame* $\mathcal{G} \cap T$ then T' is a *trap* for Odd in \mathcal{G} .

For a set of vertices $D \subseteq V$, we say that an Even *strategy* σ is an *Even dominion strategy* on D if: σ traps Odd in D and every cycle in the subgraph (D, σ) is even. Finally, we say that a set D of vertices is an Even *dominion* if there is an Even dominion strategy on it.

Odd strategies, *trapping* Even, and Odd *dominia* are defined in an analogous way by swapping the roles of the two players. It is an instructive exercise to prove the following two facts about Even and Odd *dominia*.

Proposition 1 (Closure under union). *If D and D' are Even (resp. Odd) *dominia* then $D \cup D'$ is also an Even (resp. Odd) *dominion*.*

Proposition 2 (*Dominion disjointness*). *If D is an Even *dominion* and D' is an Odd *dominion* then $D \cap D' = \emptyset$.*

From closure under union it follows that in every *parity game*, there is the largest Even *dominion* W_{Even} (which is the union of all Even *dominia*) and the largest Odd *dominion* W_{Odd} (which is the union of all Odd *dominia*), and from *dominion disjointness* it follows that the two sets are disjoint. The positional determinacy theorem states that, remarkably, the largest Even *dominion* and the largest Odd *dominion* form a partition of the set of vertices.

Theorem 3 (Positional determinacy [EJ91]). *Every vertex in a given *parity game* is either in the largest Even *dominion* or in the largest Odd *dominion*.*

2.2. REACHABILITY STRATEGIES AND ATTRACTORS

In a *parity game* \mathcal{G} , for a target set of vertices B (“bullseye”) and a set of vertices A such that $B \subseteq A$, we say that an Even *strategy* σ is an *Even reachability strategy* to B from A if every infinite path in the subgraph (V, σ) that starts from a vertex in A contains at least one vertex in B .

For every target set B , there is the largest (with respect to set inclusion) set from which there is an Even reachability **strategy** to B in \mathcal{G} ; we call this set the Even **attractor** to B in \mathcal{G} and denote it by $\text{Attr}_{\text{Even}}^{\mathcal{G}}(B)$. Odd reachability strategies and Odd **attractors** are defined analogously.

We highlight the simple facts that if A is an **attractor** for a player in \mathcal{G} then its complement $V \setminus A$ is a **trap** for her; and that attractors are monotone operators: if $B' \subseteq B$ then the **attractor** to B' is included in the **attractor** to B .

2.3. ATTRACTOR DECOMPOSITIONS

If \mathcal{G} is a **parity game** in which all **priorities** do not exceed a non-negative even number d then we say that

$$\mathcal{H} = \langle A, (S_1, \mathcal{H}_1, A_1), \dots, (S_k, \mathcal{H}_k, A_k) \rangle$$

is an Even d -**attractor decomposition** of \mathcal{G} if:

- A is the Even **attractor** to the (possibly empty) set of vertices of **priority** d in \mathcal{G} ;

and setting $\mathcal{G}_1 = \mathcal{G} \setminus A$, for all $i = 1, 2, \dots, k$, we have:

- S_i is a non-empty **trap** for Odd in \mathcal{G}_i in which every vertex **priority** is at most $d - 2$;
- \mathcal{H}_i is a $(d - 2)$ -**attractor decomposition** of **subgame** $\mathcal{G} \cap S_i$;
- A_i is the Even **attractor** to S_i in \mathcal{G}_i ;
- $\mathcal{G}_{i+1} = \mathcal{G}_i \setminus A_i$;

and the game \mathcal{G}_{k+1} is empty. If $d = 0$ then we require that $k = 0$.

The following proposition states that if a **subgame** induced by a **trap** for Odd has an Even **attractor decomposition** then the **trap** is an Even **dominion**. Indeed, a routine proof argues that the union of all the reachability strategies, implicit in the attractors listed in the decomposition, is an Even **dominion strategy**.

Proposition 4. *If d is even, T is a **trap** for Odd in \mathcal{G} , and there is an Even d -**attractor decomposition** of $\mathcal{G} \cap T$, then T is an Even **dominion** in \mathcal{G} .*

By symmetry, the dual proposition holds for player Even, assuming that d is odd.

Attractor decompositions are witnesses for the largest **dominia** and that the classic recursive **McNaughton-Zielonka algorithm** can be amended to produce such witnesses. We provide the details of this claim in Appendix A. Since **McNaughton-Zielonka algorithm** produces Even and Odd **attractor decompositions**, respectively, of **subgames** that are induced by sets of vertices that are complements of each other, a by-product of its analysis is a constructive proof of the positional determinacy theorem (Theorem 3).

Theorem 5. *McNaughton-Zielonka algorithm can be enhanced to produce both the largest Even and Odd **dominia**, and an **attractor decomposition** of each. Every vertex is in one of the two **dominia**.*

3. UNIVERSAL TREES AND ALGORITHMS

The running time of the **McNaughton-Zielonka algorithm** is, up to a small polynomial factor, determined by the number of recursive calls it makes overall. While numerous experiments indicate that the algorithm performs very well on some classes of random games and on games arising from applications in model checking, temporal logic synthesis, and equivalence checking [vD18], it is also well known that there are families of **parity**

games on which [McNaughton-Zielonka algorithm](#) performs exponentially many recursive calls [[Fri11a](#)].

Parys [[Par19](#)] has devised an ingenious modification of [McNaughton-Zielonka algorithm](#) that reduced the number of recursive calls of the algorithm to *quasi-polynomial* number $n^{O(\lg n)}$ in the worst case. Lehtinen, Schewe, and Wojtczak [[LSW19](#)] have slightly modified [Parys's algorithm](#) in order to improve the running time from $n^{O(\lg n)}$ down to $d^{O(\lg n)}$ for (n, d) -small parity games. They have also made an informal observation that the tree of recursive calls of their recursive procedure is [universal](#).

In this paper, we argue that [McNaughton-Zielonka algorithm](#), [Parys's algorithm](#), and [Lehtinen-Schewe-Wojtczak algorithm](#) are special cases of what we call a [universal attractor decomposition algorithm](#). The [universal algorithm](#) is parameterized by two ordered trees and we prove a striking structural result that if those trees are capacious enough to embed (in a formal sense explained later) ordered trees that describe the “shape” of some [attractor decompositions](#) of the largest Even and Odd [dominia](#) in a [parity game](#), then the [universal algorithm](#) correctly computes the two [dominia](#). It follows that if the algorithm is run on two [universal trees](#) then it is correct, and indeed we reproduce [McNaughton-Zielonka](#), [Parys's](#), and [Lehtinen-Schewe-Wojtczak algorithm](#) by running the [universal algorithm](#) on specific classes of [universal trees](#). In particular, [Lehtinen-Schewe-Wojtczak algorithm](#) is obtained by using the succinct [universal trees](#) of Jurdziński and Lazić [[JL17](#)], whose size nearly matches the *quasi-polynomial* lower bound on the size of [universal trees](#) [[CDF⁺19](#)].

3.1. UNIVERSAL ORDERED TREES

Ordered trees. Ordered trees are defined inductively; an ordered tree is the trivial tree $\langle \rangle$ or a sequence $\langle \mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_k \rangle$, where \mathcal{T}_i is an ordered tree for every $i = 1, 2, \dots, k$. For an ordered tree \mathcal{T} , we denote its *number of leaves* by $\text{leaves}(\mathcal{T})$ and its *height* by $\text{height}(\mathcal{T})$, with the convention that the height of the trivial tree is zero. Moreover, we denote by $\langle \mathcal{T} \rangle^n$ the ordered tree $\langle \mathcal{T}_1, \dots, \mathcal{T}_i \rangle$ where \mathcal{T}_i is a copy of \mathcal{T} for each $i = 1, 2, \dots, n$.

Trees of attractor decompositions. The definition of an [attractor decomposition](#) is inductive and we define an ordered tree that reflects the hierarchical structure of an [attractor decomposition](#). If d is even and

$$\mathcal{H} = \langle A, (S_1, \mathcal{H}_1, A_1), \dots, (S_k, \mathcal{H}_k, A_k) \rangle$$

is an Even d -[attractor decomposition](#) then we define the *tree of attractor decomposition* \mathcal{H} , denoted by $\mathcal{T}_{\mathcal{H}}$, to be the trivial ordered tree $\langle \rangle$ if $k = 0$, and otherwise, to be the ordered tree $\langle \mathcal{T}_{\mathcal{H}_1}, \mathcal{T}_{\mathcal{H}_2}, \dots, \mathcal{T}_{\mathcal{H}_k} \rangle$, where for every $i = 1, 2, \dots, k$, tree $\mathcal{T}_{\mathcal{H}_i}$ is the [tree of attractor decomposition](#) \mathcal{H}_i . Trees of Odd [attractor decompositions](#) are defined analogously.

Observe that the sets S_1, S_2, \dots, S_k in an [attractor decomposition](#) as above are non-empty and pairwise disjoint, which implies that trees of [attractor decompositions](#) are small relative to the number of vertices and the number of distinct [priorities](#) in a [parity game](#). More precisely, we say that an ordered tree is (n, h) -*small* if its height is at most h and it has at most n leaves. The following proposition can be proved by routine structural induction.

Proposition 6. *If \mathcal{H} is an [attractor decomposition](#) of an (n, d) -small parity game then its tree $\mathcal{T}_{\mathcal{H}}$ is $(n, \lceil d/2 \rceil)$ -small.*

Embedding ordered trees. Intuitively, an ordered tree *embeds* another if the latter can be obtained from the former by pruning some subtrees. More formally, every ordered tree *embeds* the trivial tree $\langle \rangle$, and $\langle \mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_k \rangle$ *embeds* $\langle \mathcal{T}'_1, \mathcal{T}'_2, \dots, \mathcal{T}'_\ell \rangle$ if there are indices i_1, i_2, \dots, i_ℓ , such that $1 \leq i_1 < i_2 < \dots < i_\ell \leq k$ and for every $j = 1, 2, \dots, \ell$, we have that \mathcal{T}_{i_j} *embeds* \mathcal{T}'_j .

Universal ordered trees. We say that an ordered tree is (n, h) -*universal* [CDF⁺19] if it *embeds* every (n, h) -*small* ordered tree. The complete n -ary tree of height h can be defined by induction on h : if $h = 0$ then $C_{n,0}$ is the trivial tree $\langle \rangle$, and if $h > 0$ then $C_{n,h}$ is the ordered tree $\langle C_{n,h-1} \rangle^n$. The tree $C_{n,h}$ is obviously (n, h) -*universal* but its size is exponential in h .

We define two further classes $P_{n,h}$ and $S_{n,h}$ of (n, h) -universal trees, introduced respectively by Parys [Par19] and by Jurdziński and Łazić [JL17], whose size is only *quasi-polynomial*, and hence they are significantly smaller than the complete n -ary trees of height h . Both classes are defined by induction on $n + h$.

If $h = 0$ then both $P_{n,h}$ and $S_{n,h}$ are defined to be the trivial tree $\langle \rangle$. If $h > 0$ then $P_{n,h}$ is defined to be the ordered tree

$$\langle P_{\lfloor n/2 \rfloor, h-1} \rangle^{\lfloor n/2 \rfloor} \cdot \langle P_{n, h-1} \rangle \cdot \langle P_{\lfloor n/2 \rfloor, h-1} \rangle^{\lfloor n/2 \rfloor},$$

and $S_{n,h}$ is defined to be the ordered tree

$$S_{\lfloor n/2 \rfloor, h} \cdot \langle S_{n, h-1} \rangle \cdot S_{\lfloor n/2 \rfloor, h}.$$

The following proposition can easily be proven by induction on (n, h) .

Proposition 7. *Ordered trees $C_{n,h}$, $P_{n,h}$ and $S_{n,h}$ are (n, h) -universal.*

A proof of *universality* of $S_{n,h}$ is implicit in the work of Jurdziński and Łazić [JL17], whose *succinct multi-counters* are merely an alternative presentation of trees $S_{n,h}$. Parys [Par19] has shown that the number of leaves in trees $P_{n,h}$ is $n^{\lg n + O(1)}$ and Jurdziński and Łazić [JL17] have proved that the number of leaves in trees $S_{n,h}$ is $n^{\lg h + O(1)}$. Czerwinski et al. [CDF⁺19] have established a *quasi-polynomial* lower bound on the number of leaves in (n, h) -*universal trees*, which the size of $S_{n,h}$ exceeds only by a small polynomial factor.

3.2. UNIVERSAL ALGORITHM

Every call of *McNaughton-Zielonka algorithm* (Algorithm 2) repeats the main loop until the set returned by a recursive call is empty. If the number of iterations for each value of d is large then the overall number of recursive calls may be exponential in d in the worst case, and that is indeed what happens for some families of hard *parity games* [Fri11a].

In our *universal attractor decomposition algorithm* (Algorithm 1), every iteration of the main loop performs exactly the same actions as in *McNaughton-Zielonka algorithm* (see Algorithm 2 and Figure 2), but the algorithm uses a different mechanism to determine how many iterations of the main loop are performed in each recursive call. In the mutually recursive procedures UnivOdd and UnivEven , this is determined by the numbers of children of the root in the input trees $\mathcal{T}^{\text{Even}}$ (the third argument) and \mathcal{T}^{Odd} (the fourth argument), respectively. Note that the sole recursive call of UnivOdd in the i -th iteration of the main loop in a call of UnivEven is given subtree $\mathcal{T}_i^{\text{Odd}}$ as its fourth argument and, analogously,


```

procedure  $\text{UnivEven}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$ :
  let  $\mathcal{T}^{\text{Odd}} = \langle \mathcal{T}_1^{\text{Odd}}, \mathcal{T}_2^{\text{Odd}}, \dots, \mathcal{T}_k^{\text{Odd}} \rangle$ 
   $\mathcal{G}_1 \leftarrow \mathcal{G}$ 
  for  $i \leftarrow 1$  to  $k$  do
     $D_i \leftarrow \pi^{-1}(d) \cap \mathcal{G}_i$ 
     $\mathcal{G}'_i \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$ 
     $\mathcal{U}_i \leftarrow \text{UnivOdd}(\mathcal{G}'_i, d-1, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$ 
     $\mathcal{G}_{i+1} \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(\mathcal{U}_i)$ 
  return  $\bigvee \mathcal{G}_{k+1}$ 

procedure  $\text{UnivOdd}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$ :
  let  $\mathcal{T}^{\text{Even}} = \langle \mathcal{T}_1^{\text{Even}}, \mathcal{T}_2^{\text{Even}}, \dots, \mathcal{T}_\ell^{\text{Even}} \rangle$ 
   $\mathcal{G}_1 \leftarrow \mathcal{G}$ 
  for  $i \leftarrow 1$  to  $\ell$  do
     $D_i \leftarrow \pi^{-1}(d) \cap \mathcal{G}_i$ 
     $\mathcal{G}'_i \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(D_i)$ 
     $\mathcal{U}_i \leftarrow \text{UnivEven}(\mathcal{G}'_i, d-1, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$ 
     $\mathcal{G}_{i+1} \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(\mathcal{U}_i)$ 
  return  $\bigvee \mathcal{G}_{\ell+1}$ 

```

ALGORITHM 1: *The universal attractor decomposition algorithm.*

the sole recursive call of UnivEven in the j -th iteration of the main loop in a call of UnivOdd is given subtree $\mathcal{T}_j^{\text{Even}}$ as its third argument.

In order to characterise the tree of recursive calls, let us define the *interleaving* operation on two ordered trees inductively as follows: $\langle \rangle \bowtie \mathcal{T} = \langle \rangle$ and $\langle \mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_k \rangle \bowtie \mathcal{T} = \langle \mathcal{T} \bowtie \mathcal{T}_1, \mathcal{T} \bowtie \mathcal{T}_2, \dots, \mathcal{T} \bowtie \mathcal{T}_k \rangle$. Then the following simple proposition provides an explicit description of the tree of recursive calls of our *universal algorithm*. We state it only for the case where d is even, but a similar proposition holds when d is odd if trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} are swapped in the statement.

Proposition 8. *If d is even then the tree of recursive calls to the procedure $\text{UnivEven}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$ is the interleaving $\mathcal{T}^{\text{Odd}} \bowtie \mathcal{T}^{\text{Even}}$ of trees \mathcal{T}^{Odd} and $\mathcal{T}^{\text{Even}}$.*

The following elementary proposition helps estimate the size of an *interleaving* of two ordered trees and hence the running time of a call of the *universal algorithm* that is given two ordered trees as inputs.

Proposition 9. *If \mathcal{T} and \mathcal{T}' are ordered trees then:*

- $\text{height}(\mathcal{T} \bowtie \mathcal{T}') \leq \text{height}(\mathcal{T}) + \text{height}(\mathcal{T}')$;
- $\text{leaves}(\mathcal{T} \bowtie \mathcal{T}') \leq \text{leaves}(\mathcal{T}) \cdot \text{leaves}(\mathcal{T}')$.

In contrast to the *universal algorithm*, the tree of recursive calls of *McNaughton-Zielonka algorithm* is not pre-determined by a structure separate from the game graph, such as the pair of trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} . Instead, *McNaughton-Zielonka algorithm* determines the number of iterations of its main loop adaptively, using the *adaptive empty-set early termination rule*: terminate the main loop as soon as $\mathcal{U}_i = \emptyset$. We argue that if we add the *empty-set early termination rule* to the *universal algorithm* in which both trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} are the tree $C_{n,d/2}$ then its behaviour coincides with *McNaughton-Zielonka algorithm*.

Proposition 10. *The [universal algorithm](#) performs the same actions and produces the same output as [McNaughton-Zielonka algorithm](#) if it is run on an (n, d) -small parity game and with both trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} equal to $C_{n,d/2}$, and if it uses the [adaptive empty-set early termination rule](#).*

The idea of using rules for implicitly pruning the tree of recursive calls of a [McNaughton-Zielonka](#)-style algorithm that are significantly different from the [adaptive empty-set early termination rule](#) is due to Parys [Par19]. In this way, he has designed the first [McNaughton-Zielonka](#)-style algorithm that works in [quasi-polynomial](#) time $n^{O(\lg n)}$ in the worst case, and Lehtinen, Schewe, and Wojtczak [LSW19] have refined [Parys’s algorithm](#), improving the worst-case running time down to $n^{O(\lg d)}$. Both algorithms use two numerical arguments (one for Even and one for Odd) and “halving tricks” on those parameters, which results in pruning the tree of recursive calls down to [quasi-polynomial](#) size in the worst case. We note that our [universal algorithm](#) yields the algorithms of Parys and of Lehtinen et al., respectively, if, when run on an (n, d) -small parity game and if both trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} set to be the $(n, d/2)$ -universal trees $P_{n,d/2}$ and $S_{n,d/2}$, respectively.

Proposition 11. *The [universal algorithm](#) performs the same actions and produces the same output as [Lehtinen-Schewe-Wojtczak algorithm](#) if it is run on an (n, d) -small parity game with both trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} equal to $S_{n,d/2}$.*

The correspondence between the [universal algorithm](#) executed on $(n, d/2)$ -universal trees $P_{n,d/2}$ and [Parys’s algorithm](#) is a bit more subtle. While both run in quasi-polynomial time in the worst case, the former may perform more recursive calls than the latter. The two coincide, however, if the former is enhanced with a simple adaptive tree-pruning rule similar to the [empty-set early termination rule](#). The discussion of this and other adaptive tree-pruning rules will be better informed once we have discussed sufficient conditions for the correctness of our [universal algorithm](#). Therefore, we will return to elaborating the full meaning of the following proposition in Section 4.4.

Proposition 12. *The [universal algorithm](#) performs the same actions and produces the same output as a non-adaptive version of [Parys’s algorithm](#) if it is run on an (n, d) -small parity games with both trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} equal to $P_{n,d/2}$.*

4. CORRECTNESS VIA STRUCTURAL THEOREMS

The classical proof of the correctness of [McNaughton-Zielonka algorithm](#) [AG11] essentially relies on claim that when one reaches the [empty-set condition](#), then this proves that we’ve precisely computed the opponent’s winning region. The argument breaks down if the loop terminates before that [empty-set condition](#) obtains. Instead, Parys [Par19] has developed a novel *dominion separation technique* to prove correctness of his algorithm and Lehtinen et al. [LSW19] use the same technique to justify theirs.

In this paper, we significantly generalize the [dominion separation](#) technique of Parys, which allows us to intimately link the correctness of our meta-algorithm to shapes (modelled as ordered trees) of [attractor decompositions](#) of largest Even and Odd dominia. We say that the [universal algorithm](#) is correct on a parity game if $\text{Univ}_{\text{Even}}$ returns the largest Even dominion and Univ_{Odd} returns the largest Odd dominion. We also say that an ordered tree \mathcal{T} *embeds* a dominion D in a parity game \mathcal{G} if it *embeds* the tree of some [attractor decomposition](#) of $\mathcal{G} \cap D$. The main technical result we aim to prove in this section is the sufficiency of the following condition for the [universal algorithm](#) to be correct.

Theorem 13 (Correctness of universal algorithm). *The [universal algorithm](#) is correct on a*

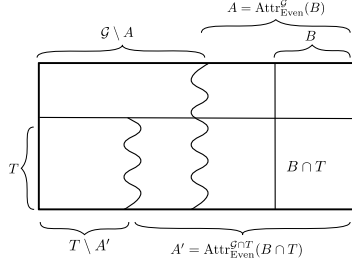


FIGURE 1: *Traps and attractors in Proposition 16.*

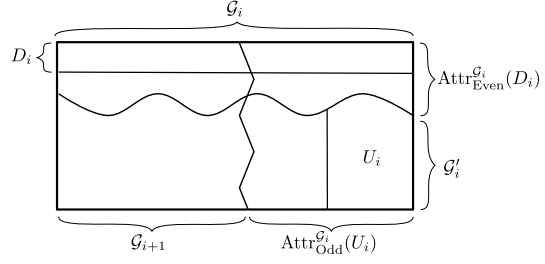


FIGURE 2: *Attractors and subgames in one iteration of the loop in attractor decomposition algorithms.*

parity game \mathcal{G} if it is run on ordered trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} , such that $\mathcal{T}^{\text{Even}}$ embeds the largest Even dominion in \mathcal{G} and \mathcal{T}^{Odd} embeds the largest Odd dominion in \mathcal{G} .

4.1. EMBEDDABLE DECOMPOSITION THEOREM

Before we prove Theorem 13, in this section we establish another technical result—the *embeddable decomposition theorem*—that enables our generalization of Parys’s dominion separation technique. Its statement is intuitive: a subgame induced by a trap has a simpler attractor decomposition structure than the whole game itself; its proof, however, seems to require some careful surgery.

Theorem 14 (Embeddable decomposition). *If T is a trap for Even in a parity game \mathcal{G} and $\mathcal{G}' = \mathcal{G} \cap T$ is the subgame induced by T , then for every Even attractor decomposition \mathcal{H} of \mathcal{G} , there is an Even attractor decomposition \mathcal{H}' of \mathcal{G}' , such that $\mathcal{T}_{\mathcal{H}}$ embeds $\mathcal{T}_{\mathcal{H}'}$.*

See the proof of Theorem 14 at page 21.

In order to streamline the proof of the embeddable decomposition theorem, we state the following two propositions, which synthesize or generalize some of the arguments that were also used by Lehtinen, Parys, Schewe and Wojtczak [LPSW22]. Proofs are included in the Appendix.

Proposition 15. *Suppose that R is a trap for Even in game \mathcal{G} . Then if T is a trap for Odd in \mathcal{G} then $T \cap R$ is a trap for Odd in subgame $\mathcal{G} \cap R$, and if T is an Even dominion in \mathcal{G} then $T \cap R$ is an Even dominion in $\mathcal{G} \cap R$.*

The other proposition is illustrated in Figure 1. Its statement is more complex than that of the first proposition. The statement and the proof describe the relationship between the Even attractor of a set B of vertices in a game \mathcal{G} and the Even attractor of the set $B \cap T$ in subgame $\mathcal{G} \cap T$, where T is a trap for Even in \mathcal{G} .

Proposition 16. *Let $B \subseteq V^{\mathcal{G}}$ and let T be a trap for Even in game \mathcal{G} . Define $A = \text{Attr}_{\text{Even}}^{\mathcal{G}}(B)$ and $A' = \text{Attr}_{\text{Even}}^{\mathcal{G} \cap T}(B \cap T)$. Then $T \setminus A'$ is a trap for Even in subgame $\mathcal{G} \setminus A$.*

We prove the embeddable decomposition theorem by induction on the number of leaves of the tree of attractor decomposition \mathcal{H} . Note that our definition of an attractor decomposition allows for S_i to be any non-empty trap for Odd in \mathcal{G}_i in which every vertex priority is at most $d - 2$, whereas Daviaud, Jurdziński, and Lehtinen’s definition [DJL19] ask for S_i to be the maximal trap for Odd satisfying the aforementioned property. Relaxing the definition of attractor decompositions is crucial for Proposition 16 to hold.

4.2. DOMINION SEPARATION THEOREM

The simple dominion disjointness property (Proposition 2) states that every Even **dominion** is disjoint from every Odd **dominion**. For two sets A and B , we say that another set X *separates* A from B if $A \subseteq X$ and $X \cap B = \emptyset$. In this section we establish a very general **dominion separation property** for **subgames** that occur in iterations of the **universal algorithm**. This allows us to prove one of the main technical results of this paper (Theorem 13) that describes a detailed structural sufficient condition for the correctness of the **universal algorithm**.

Theorem 17 (Dominion separation). *Let \mathcal{G} be an (n, d) -small parity game and let $\mathcal{T}^{\text{Even}} = \langle \mathcal{T}_1^{\text{Even}}, \dots, \mathcal{T}_\ell^{\text{Even}} \rangle$ and $\mathcal{T}^{\text{Odd}} = \langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_k^{\text{Odd}} \rangle$ be trees of height at most $\lceil d/2 \rceil$ and $\lfloor d/2 \rfloor$, respectively.*

See the proof of Theorem 17 at page 23.

- (a) *If d is even and $\mathcal{G}_1, \dots, \mathcal{G}_{k+1}$ are the games that are computed in the successive iterations of the loop in the call $\text{UnivEven}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$, then for every $i = 0, 1, \dots, k$, we have that \mathcal{G}_{i+1} separates every Even **dominion** in \mathcal{G} that tree $\mathcal{T}^{\text{Even}}$ *embeds* from every Odd **dominion** in \mathcal{G} that tree $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_i^{\text{Odd}} \rangle$ *embeds*.*
- (b) *If d is odd and $\mathcal{G}_1, \dots, \mathcal{G}_{\ell+1}$ are the games that are computed in the successive iterations of the loop in the call $\text{UnivOdd}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$, then for every $i = 0, 1, \dots, \ell$, we have that \mathcal{G}_{i+1} separates every Odd **dominion** in \mathcal{G} that tree \mathcal{T}^{Odd} *embeds* from every Even **dominion** in \mathcal{G} that tree $\langle \mathcal{T}_1^{\text{Even}}, \dots, \mathcal{T}_i^{\text{Even}} \rangle$ *embeds*.*

4.3. CORRECTNESS AND COMPLEXITY

The **dominion separation theorem** (Theorem 17) allows us to conclude the proof of the main **universal algorithm** correctness theorem (Theorem 13). Indeed, if trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} satisfy the conditions of Theorem 13 then, by the **dominion separation theorem**, the set returned by the call $\text{UnivEven}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$ separates the largest Even **dominion** from the largest Odd **dominion**, and hence—by the positional determinacy theorem (Theorem 3)—it is the largest Even dominion. The argument for procedure UnivOdd is analogous.

We note that the **universal algorithm** correctness theorem, together with Propositions 12 and 11, imply correctness of the non-adaptive version of Parys’s algorithm [Par19] and of Lehtinen-Schewe-Wojtczak algorithm [LSW19], because trees of attractor decompositions are $(n, d/2)$ -small (Proposition 6) and trees $P_{n, d/2}$ and $S_{n, d/2}$ are $(n, d/2)$ -universal.

The following fact, an alternative restatement of the conclusion of Lehtinen et al. [LSW19], is a simple corollary of the precise asymptotic upper bounds on the size of the **universal trees** $S_{n, d/2}$ established by Jurdziński and Lazić [JL17], and of Propositions 11, 8, and 9.

Proposition 18 (Complexity). *The **universal algorithm** that uses **universal trees** $S_{n, d/2}$ (aka. Lehtinen-Schewe-Wojtczak algorithm) solves (n, d) -small parity games in polynomial time if $d = O(\log n)$, and in time $n^{2 \lg(d/\lg n) + O(1)}$ if $d = \omega(\log n)$.*

4.4. ACCELERATION BY TREE PRUNING

As we have discussed in Section 3.2, Parys [Par19] has achieved a breakthrough of developing the first **quasi-polynomial McNaughton-Zielonka-style algorithm** for **parity games** by pruning the tree of recursive calls down to **quasi-polynomial** size. Proposition 12 clarifies that Parys’s scheme can be reproduced by letting the **universal algorithm** run on **universal**

trees $P_{n,d/2}$, but as it also mentions, just doing so results in a “non-adaptive” version of Parys’s algorithm. What is the “adaptive” version actually proposed by Parys?

Recall that the root of tree $P_{n,h}$ has $n + 1$ children, the first $n/2$ and the last $n/2$ children are the roots of copies of tree $P_{n/2,h-1}$, and the middle child is the root of a copy of tree $P_{n,h-1}$. The adaptive version of Parys’s algorithm also uses another tree-pruning rule, which is adaptive and a slight generalization of the empty-set rule: whenever the algorithm is processing the block of the first $n/2$ children of the root or the last $n/2$ children of the root, if one of the recursive calls in this block returns an empty set then the rest of the block is omitted.

We expect that our structural results (such as Theorems 13 and 17) will provide insights to inspire development and proving correctness of further and more sophisticated adaptive tree-pruning rules, but we leave it to future work. This may be critical for making quasi-polynomial versions of McNaughton-Zielonka competitive in practice with its basic version that is exponential in the worst case, but remains very hard to beat in practice [vD18, LPSW22].

5. COMPUTING NESTED FIXPOINTS

Computing fixpoints is fundamental in the study of computer science. Solving nested fixpoint equations (NFEs) over finite lattices are known to be computationally equivalent to solving parity games [BKMP19], however, most of the reductions involve an exponential increase in the size of the resulting parity game. The satisfiability problem of the coalgebraic μ -calculus has also been reduced to the same [HS19]. A corollary of Calude et. al’s breakthrough result was that specific kinds of fixpoint equations could be solved in quasi-polynomial time. Following this progress, there were several algorithms targeted at solving more general fixpoint equations by using universal graphs [HS21] and universal trees [ANP21]. Hausman and Schröder gave a quasi-polynomial algorithm to solve NFEs using progress measures on universal graphs whereas Arnold, Niwinski and Parys solved NFEs using the key result on decompositions of dominia similar to an earlier version of this paper. Here, we provide a slightly different way of solving nested fixpoints by converting the equation to an exponentially sized fixpoint game, as in [HS21] but using our universal attractor decomposition algorithm, parameterised by two trees, as in [ANP21]. The algorithm proposed by Arnold, Niwinski and Parys is similar to ours, in the sense that both algorithms use a pair of trees to guide the computation of a subset of a complete lattice and of a set of vertices in a parity game in our case, respectively. Since we can describe the set of winning vertices for some player in a parity game with a formula whose length is linear in the number of distinct priorities: d , the algorithm of [ANP21] can be seen as a generalisation of Algorithm 1. On the other hand, we explain in this section how to, given a nested fixpoint equation, run the latter algorithm on a parity game—called a fixpoint game—, which has an exponential size compared to the size of the NFE, without having an exponential blowup. We thus obtain an algorithm to compute nested fixpoint equations in quasi-polynomial time. In this sense, we argue that the algorithm of Arnold, Niwinski and Parys is equivalent to ours. However, it should be noted that [ANP21] provides an asymmetrical version of their algorithm—using a technique of Seidl [Sei96]—which is quadratically faster, in the worse case, than Algorithm 1. In section 6, a detailed description of how to implement symbolically both variants of the universal algorithm, for parity games and nested fixpoint, which require logarithmically less symbolic space than Chatterjee, Dvořák, Henzinger and Svozil quasi-polynomial symbolic algorithm [CDHS18]

is provided.

We argue that we can directly apply our [universal attractor decomposition algorithm](#) on these exponential sized [fixpoint games](#) with the help of a carefully designed data structure, which ensures that we can in fact compute fixpoints using our algorithm in time proportional to $|\mathcal{T}_{\text{Odd}}| \cdot |\mathcal{T}_{\text{Even}}|$.

5.1. NESTED FIXPOINT EQUATIONS

In this subsection, we will define [nested fixpoint equations](#) over the powerset lattice. Consider a finite set of elements U and its powerset lattice $\mathcal{P}(U)$. Let f be a monotone function (component wise) from $\mathcal{P}(U)^d$ to $\mathcal{P}(U)^d$. The function f can be expressed as a tuple (f_1, \dots, f_d) of functions from $\mathcal{P}(U)^d$ to $\mathcal{P}(U)$, where f_i is the projection of f to the i -th component.

Since there is a natural bijection from d tuples of subsets of U to subsets of $(U \times [d])$, we instead denote f as a function from $\mathcal{P}(U \times [d])$ to $\mathcal{P}(U \times [d])$.

A [nested fixpoint equation](#) is a system of d fixpoint equations of the form:

$$X_i =_{\eta_i} f_i(X_1, \dots, X_d) \quad (*)$$

for i ranging from $1, \dots, d$ and where $\eta_i = \nu$ if i is even, and $\eta_i = \mu$ otherwise. We refer to a system such as $(*)$ as a [nested fixpoint equation](#) and refer to it with the short hand: $X =_{\eta} f(X)$. One could consider a more general form of [fixpoint equations](#) where $\eta_i \in \{\mu, \nu\}$, but for simplicity of presentation, we restrict ourselves to the above.

The [solution](#) of a system of d [fixpoint equations](#) as the one defined by $(*)$, is a subset of $U \times [d]$, defined recursively as follows. We say that the solution of the empty set of equations is the empty tuple. For a system of one or more fixpoint equations, we define a function f^{d-1} from subsets of U to subsets of $(U \times [d-1])$. This function f^{d-1} takes as input Y_d , a subset of U , and uses this input to fix $X_d = Y_d$ in the system of equations and the solution obtained to the system of $d-1$ equations by fixing X_d to be Y_d is the output of f^{d-1} . We finally say the solution of the system of equations is $(f^{d-1}(Y_d), Y_d)$, where $Y_d = \eta_d(\lambda X_d. f_d(f^{d-1}(X_d), X_d))$.

5.2. FIXPOINT GAMES

Let us now define an equivalent [parity game](#) \mathcal{G}_f , called a [fixpoint game](#). Solving the parity game \mathcal{G}_f correlates to finding the [solution](#) of the system of [nested fixpoint equation](#) defined by $X =_{\eta} f(X)$ [BKMM19, HS21].

Here, $\mathcal{G}_f = (V_f, E_f)$ with the [priority function](#) π_f , where V_f consists of the disjoint union $(U \times [d]) \cup \{v_A \mid A \subseteq U \times [d]\}$. The vertices corresponding to elements of the set $(U \times [d])$ belong to Even and the ones corresponding to subsets of the same set belong to Odd. The priority function π_f assigns Even's vertices (u, i) to i , and vertices Odd's vertices to priority 0. The edges from a vertex (u, i) belonging to Even in \mathcal{G}_f lead to the set of Odd vertices $\{v_A \mid (u, i) \in f(A)\}$ and edges from a vertex v_A , belonging to Odd lead to the set of Even vertices $\{(u, i) \mid (u, i) \in A\}$.

Finding if (u, i) is in the [solution](#) of a [nested fixpoint equation](#) $X =_{\eta} f(X)$ is known to be equivalent to solving the corresponding [fixpoint game](#) \mathcal{G}_f of the equation from the even vertex (u, i) , as shown in Theorem 4.8 of [BKMM19].

5.3. SOLVING FIXPOINT GAMES

We provide a way to solve a **fixpoint game** with the help of the **universal attractor decomposition algorithm** in Section 3.2.

We define a specific kind of **subgames** that we call **flowery subgames** and show that they are pertinent to solving **fixpoint games** using the **universal attractor decomposition algorithm**. Given two subsets $\emptyset \subsetneq Y \subseteq X \subseteq \mathcal{U} \times [d]$, we define the flowery subgame on (X, Y) , denoted by $\mathcal{F}(X, Y)$, to be the **subgame** of \mathcal{G}_f whose set of vertices consists of all Odd vertices v_A which is a subset of X intersecting non-trivially with Y , resembling the petal of a flower along with all vertices of Even belonging to Y , resembling the core of a flower. More formally, we define

$$\mathcal{F}(X, Y) = Y \uplus \{v_A \mid A \subseteq X \text{ and } A \cap Y \neq \emptyset\}.$$

In the game \mathcal{G}_f , on removing vertices that have no outgoing edges along with the respective **attractors** to these sets of vertices, i.e., Odd **attractors** to Even vertices with no outgoing edges and vice versa, we get a **flowery subgame**. Moreover, the following lemma reassures us that all significant operations performed by Algorithm 1 on flowery subgames, results in flowery subgames.

Lemma 19 (Floweriness). *If $\text{Univ}_{\text{Even}}$ (resp., Univ_{Odd}) is run on a **flowery subgame**, for all iterations in the for-loop, **subgame** \mathcal{G}_i is also **flowery**. In particular, \mathcal{G}_{k+1} , which is the subgame returned, is **flowery**.*

See the proof of Lemma 19 at page 28.

The **attractor** to a set of vertices during a run of the algorithm can be computed by at most $d|\mathcal{U}|$ many computation of f on subsets of $\mathcal{U} \times [d]$. We can therefore solve nested **fixpoint games** in quasi-polynomial time using the **universal attractor decomposition algorithm**, by only keeping track of the sets X and Y representing each subgame, as stated below.

Theorem 20. *The modified **universal algorithm** that computes **nested fixpoint equations** on trees \mathcal{T}_{Odd} and $\mathcal{T}_{\text{Even}}$ makes $|\mathcal{T}_{\text{Odd}}| \cdot |\mathcal{T}_{\text{Even}}|$ many recursive calls. Each recursive call makes at most $2d|\mathcal{U}|$ many function evaluations of f .*

5.4. CONCURRENT PARITY GAMES

Concurrent parity games have been well studied before. We consider the two player version as studied by Chatterjee, Alfredo and Henzinger in [CAH11]. These games are played among two players—Even and Odd, but instead of partitioning the vertices among the two players, they take simultaneous actions at each vertex and the token moves to a neighbour depending on the actions of both players. One might also consider a stochastic version where the simultaneous actions are decided by a pre-decided probability distribution. Both the players are allowed to use a randomised **strategy**, i.e., a strategy where the next action is proposed with the help of a probability distribution. A state is called **limit-winning** for Even (resp. Odd) if Even (resp. Odd) has a strategy to win from that state with probability arbitrarily close to 1. The decision question we have at hand, is to determine if a state is a **limit-winning** state for a given input player. **Concurrent parity games** vary from original **parity games** in that, a player might need both infinite memory and randomisation to win these games. We refer the readers to the work of Chatterjee, Alfaro, and Henzinger [CAH11] for a rigorous definition of the above games along with examples for the claims above. In their paper, they show that solving **concurrent parity games** is in $\text{NP} \cap \text{co-NP}$ as a corollary of the following theorem.

Theorem 21 ([CAH11, Theorem 5, Lemma 29 and Lemma 30]). *Limit-winning in a concurrent parity game can be expressed as an NFE over the powerset lattice of the set of edges with alternation depth at most $2d$ for a function, whose evaluation involves solving another NFE also with depth at most $2d$.*

An easy corollary from Theorem 20 along with Theorem 21, we have the following.

Corollary 22. *Limit-winning in concurrent parity games can be solved in quasi-polynomial time.*

6. SYMBOLIC ALGORITHMS

Parity games that arise in applications, for example from the automata-theoretic model checking approaches to verification and automated synthesis, often suffer from the *state-space explosion problem*: the sizes of models are exponential (or worse) in the sizes of natural descriptions of the modelled objects, and hence the models obtained may be too large to store them explicitly in memory. One method of overcoming this problem that has been successful in the practice of algorithmic formal methods is to represent the models symbolically rather than explicitly, and to develop algorithms for solving the models that work directly on such succinct symbolic representations [BCM⁺92].

We adopt the *set-based symbolic model of computation* that was already considered for parity games by Chatterjee, Dvořák, Henzinger, and Svozil [CDHS18]. In this model, any standard computational operations on any standard data structures are allowed, but there are also the following symbolic resources available: *symbolic set variables* can be used to store sets of vertices in the graph of a parity game; basic set-theoretic operations on *symbolic set variables* are available as *primitive symbolic operations*; the *controllable predecessors operations* are available as *primitive symbolic operations*: the Even (resp. Odd) controllable predecessor, when applied to a symbolic set variable X , returns the set of vertices from which Even (resp. Odd) can force to move into the set X , by taking just one outgoing edge. Since *symbolic set variables* can represent possibly very large and complex objects, they should be treated as a costly resource.

Chatterjee et al. [CDHS18] have given a symbolic set-based algorithm that on (n, d) -small parity games uses $O(d \log n)$ of *symbolic set variables* and runs in quasi-polynomial time. While the dependence on n is only logarithmic, a natural question is whether this dependence is inherent. Given that n can be prohibitively large in applications, reducing dependence on n is desirable. In this section we argue that it is not only possible to eliminate the dependence on n entirely, but it is also possible to exponentially improve the dependence on d , resulting in a quasi-polynomial *symbolic algorithm* for solving parity games that uses only $O(\lg d)$ *symbolic set variables*.

In the *set-based symbolic model of computation*, it is routine to compute the *attractors* efficiently: it is sufficient to iterate the controllable predecessor operations. Using the results of Jurdziński and Lazić [JL17], one can also represent a path of nodes from the root to a leaf in the tree $S_{n,d/2}$ in $O(\lg n \cdot \lg d)$ bits, and for every node on such a path, to compute its number of children in $O(\lg n \cdot \lg d)$ standard primitive operations. This allows to run the whole *universal algorithm* (Algorithm 1) on an (n, d) -small parity game and two copies of trees $S_{n,d/2}$, using only $O(\lg n \cdot \lg d)$ bits to represent the relevant nodes in the trees $\mathcal{T}^{\text{Even}}$ and \mathcal{T}^{Odd} throughout the execution.

The depth of the tree of recursive calls of the *universal algorithm* on an (n, d) -small parity game is at most d . Moreover, in every recursive call, only a small constant number

of set variables is needed because only the latest sets $V_i^{G_i}$, D_i , $V_i^{G'_i}$, and U_i are needed at any time. It follows that the overall number of [symbolic set variables](#) needed to run the [universal algorithm](#) is $O(d)$. Also note that every recursive call can be implemented symbolically using a constant number of [primitive symbolic operations](#) and two symbolic [attractor](#) computations.

This improves the [symbolic space](#) from Chatterjee, Dvořák, Henzinger, and Svozil's $O(d \lg n)$ to $O(d)$, while keeping the running time quasi-polynomial. This [symbolic algorithm](#) is very simple and straightforward to implement, which makes it particularly promising and attractive for empirical evaluation and deployment in applications.

Theorem 23. *There exists a [symbolic algorithm](#) that solves (n, d) -small parity games using $O(\lg d)$ [symbolic set variables](#), $O(\log d \cdot \log n)$ bits of conventional space, and whose running time is polynomial if $d = O(\log n)$, and [quasi-polynomial](#), namely $n^{2 \lg(d/\lg n) + O(1)}$, if $d = \omega(\log n)$.*

See the proof of Theorem 23 at page 29.

Using the same arguments, we obtain a [symbolic algorithm](#) to solve [nested fixpoint equations](#) in quasi-polynomial time and $O(\lg d)$ [symbolic space](#).

7. REFERENCES

- [AG11] Krzysztof R. Apt and Erich Grädel, editors. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011. URL: <http://www.cambridge.org/gb/knowledge/isbn/item5760379>.
- [ANP21] André Arnold, Damian Niwiński, and Paweł Parys. A quasi-polynomial black-box algorithm for fixed point evaluation. In *CSL*, volume 183 of *LIPIcs*, pages 9:1–9:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.9.
- [BCM⁺92] Jerry R. Burch, Edmund M. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Inf. Comput.*, 98(2):142–170, 1992. doi:10.1016/0890-5401(92)90017-A.
- [BKMP19] Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games on continuous lattices. *Proc. ACM Program. Lang.*, 3(POPL), January 2019. doi:10.1145/3290339.
- [BW18] Julian C. Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 871–919. Springer, 2018. doi:10.1007/978-3-319-10575-8_26.
- [CAH11] Krishnendu Chatterjee, Luca De Alfaro, and Thomas A. Henzinger. Qualitative concurrent parity games. *ACM Trans. Comput. Logic*, 12(4), July 2011. doi:10.1145/1970398.1970404.
- [CDF⁺19] Wojciech Czerwiński, Laure Daviaud, Nathanaël Fijalkow, Marcin Jurdziński, Ranko Lazić, and Paweł Parys. Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In *SODA*, pages 2333–2349, 2019. doi:10.1137/1.9781611975482.142.
- [CDHS18] Krishnendu Chatterjee, Wolfgang Dvořák, Monika Henzinger, and Alexander Svozil. Quasipolynomial set-based symbolic algorithms for parity games. In *LPAR-22*, volume 57 of *EPiC Series in Computing*, pages 233–253, 2018. doi:10.29007/5z5k.

- [CJK⁺17] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *STOC*, pages 252–263, 2017. doi:[10.1145/3055399.3055409](https://doi.org/10.1145/3055399.3055409).
- [DJL18] Laure Daviaud, Marcin Jurdziński, and Ranko Lazić. A pseudo-quasi-polynomial algorithm for mean-payoff parity games. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 325–334. ACM, 2018. doi:[10.1145/3209108.3209162](https://doi.org/10.1145/3209108.3209162).
- [DJL19] Laure Daviaud, Marcin Jurdziński, and Karoliina Lehtinen. Alternating weak automata from universal trees. In *CONCUR 2019, August 27-30, 2019, LIPIcs*, pages 18:1–18:14, 2019. doi:[10.4230/LIPIcs.CONCUR.2019.18](https://doi.org/10.4230/LIPIcs.CONCUR.2019.18).
- [DJT20] Laure Daviaud, Marcin Jurdziński, and K. S. Thejaswini. The strahler number of a parity game. In *ICALP 2020, July 8-11, 2020*, volume 168 of *LIPIcs*, pages 123:1–123:19, 2020. doi:[10.4230/LIPIcs.ICALP.2020.123](https://doi.org/10.4230/LIPIcs.ICALP.2020.123).
- [EJ91] E. Allen Emerson and Charanjit Jutla. Tree automata, mu-calculus and determinacy. In *FOCS*, pages 368–377, 1991. doi:[10.1109/SFCS.1991.185392](https://doi.org/10.1109/SFCS.1991.185392).
- [EJS93] E. Allen Emerson, Charanjit S. Jutla, and Aravinda Prasad Sistla. On model-checking for fragments of μ -calculus. In Costas Courcoubetis, editor, *Computer Aided Verification, 5th International Conference, CAV '93, Elounda, Greece, June 28 - July 1, 1993, Proceedings*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer, 1993. doi:[10.1007/3-540-56922-7_32](https://doi.org/10.1007/3-540-56922-7_32).
- [EWS05] Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. *SIAM J. Comput.*, 34(5):1159–1175, 2005. doi:[10.1137/S0097539703420675](https://doi.org/10.1137/S0097539703420675).
- [Fea10] John Fearnley. Exponential lower bounds for policy iteration. In *ICALP*, volume 6199 of *Lecture Notes in Computer Science*, pages 551–562. Springer, 2010. doi:[10.1007/978-3-642-14162-1_46](https://doi.org/10.1007/978-3-642-14162-1_46).
- [FHZ11] Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *STOC*, pages 283–292. ACM, 2011. doi:[10.1145/1993636.1993675](https://doi.org/10.1145/1993636.1993675).
- [Fri09] Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. In *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS 2009, 11-14 August 2009, Los Angeles, CA, USA*, pages 145–156. IEEE Computer Society, 2009. doi:[10.1109/LICS.2009.27](https://doi.org/10.1109/LICS.2009.27).
- [Fri11a] Oliver Friedmann. Recursive algorithm for parity games requires exponential time. *RAIRO Theor. Informatics Appl.*, 45(4):449–457, 2011. doi:[10.1051/ita/2011124](https://doi.org/10.1051/ita/2011124).
- [Fri11b] Oliver Friedmann. A subexponential lower bound for Zadeh’s pivoting rule for solving linear programs and games. In *IPCO*, volume 6655 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2011. doi:[10.1007/978-3-642-20807-2_16](https://doi.org/10.1007/978-3-642-20807-2_16).
- [GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar]*,

- February 2001], volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. doi:[10.1007/3-540-36387-4](https://doi.org/10.1007/3-540-36387-4).
- [HS19] Daniel Hausmann and Lutz Schröder. Optimal satisfiability checking for arithmetic μ -calculi. In *Foundations of Software Science and Computation Structures*, pages 277–294. Springer International Publishing, 2019. doi:[10.1007/978-3-030-17127-8_16](https://doi.org/10.1007/978-3-030-17127-8_16).
- [HS21] Daniel Hausmann and Lutz Schröder. Quasipolynomial computation of nested fixpoints. In *TACAS*, volume 12651 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2021. doi:[10.1007/978-3-030-72016-2_3](https://doi.org/10.1007/978-3-030-72016-2_3).
- [HSC16] Ichiro Hasuo, Shunsuke Shimizu, and Corina Cîrstea. Lattice-theoretic progress measures and coalgebraic model checking. In *POPL*, pages 718–732. ACM, 2016. doi:[10.1145/2837614.2837673](https://doi.org/10.1145/2837614.2837673).
- [JL17] Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *LICS*, pages 1–9, 2017. doi:[10.1109/LICS.2017.8005092](https://doi.org/10.1109/LICS.2017.8005092).
- [JPZ08] Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008. doi:[10.1137/070686652](https://doi.org/10.1137/070686652).
- [Leh18] Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *LICS*, pages 639–648, 2018. doi:[10.1145/3209108.3209115](https://doi.org/10.1145/3209108.3209115).
- [LMS20] Michael Luttenberger, Philipp J. Meyer, and Salomon Sickert. Practical synthesis of reactive systems from LTL specifications via parity games. *Acta Informatica*, 57(1-2):3–36, 2020. doi:[10.1007/s00236-019-00349-3](https://doi.org/10.1007/s00236-019-00349-3).
- [LPSW22] Karoliina Lehtinen, Paweł Parys, Sven Schewe, and Dominik Wojtczak. A Recursive Approach to Solving Parity Games in Quasipolynomial Time. *Logical Methods in Computer Science*, Volume 18, Issue 1, January 2022. URL: <https://lmcs.episciences.org/8953>, doi:[10.46298/lmcs-18\(1:8\)2022](https://doi.org/10.46298/lmcs-18(1:8)2022).
- [LSW19] Karoliina Lehtinen, Sven Schewe, and Dominik Wojtczak. Improving the complexity of Parys’ recursive algorithm. arXiv:1904.11810, April 2019. doi:[10.48550/arXiv.1904.11810](https://doi.org/10.48550/arXiv.1904.11810).
- [McN93] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993. doi:[10.1016/0168-0072\(93\)90036-D](https://doi.org/10.1016/0168-0072(93)90036-D).
- [Par19] Paweł Parys. Parity games: Zielonka’s algorithm in quasi-polynomial time. In *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019*, 2019. doi:[10.4230/LIPIcs.MFCS.2019.10](https://doi.org/10.4230/LIPIcs.MFCS.2019.10).
- [Sei96] Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303–308, 1996. doi:[10.1016/0020-0190\(96\)00130-5](https://doi.org/10.1016/0020-0190(96)00130-5).
- [vD18] Tom van Dijk. Oink: An implementation and evaluation of modern parity game solvers. In *Tools and Algorithms for the Construction and Analysis of Systems, 24th International Conference, TACAS 2018*, volume 10805 of *LNCS*, pages 291–308, Thessaloniki, Greece, 2018. Springer. doi:[10.1007/978-3-319-89960-2_16](https://doi.org/10.1007/978-3-319-89960-2_16).

- [VJ00] Jens Vöge and Marcin Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *CAV 2000*, volume 1855 of *LNCS*, pages 202–215, Chicago, IL, USA, 2000. Springer. doi:[10.1007/10722167_18](https://doi.org/10.1007/10722167_18).
- [Zie98] Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998. doi:[10.1016/S0304-3975\(98\)00009-7](https://doi.org/10.1016/S0304-3975(98)00009-7).

A. McNAUGHTON-ZIELONKA ALGORITHM

```

procedure McN-ZEven( $\mathcal{G}, d$ ):
  if  $d = 0$  then
     $\perp$  return  $V^{\mathcal{G}}$ 
   $i \leftarrow 0$ ;  $\mathcal{G}_1 \leftarrow \mathcal{G}$ 
  repeat
     $i \leftarrow i + 1$ 
     $D_i \leftarrow \pi^{-1}(d) \cap \mathcal{G}_i$ 
     $\mathcal{G}'_i \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$ 
     $U_i \leftarrow \text{McN-Z}_{\text{Odd}}(\mathcal{G}'_i, d - 1)$ 
     $\mathcal{G}_{i+1} \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(U_i)$ 
  until  $U_i = \emptyset$ 
   $\perp$  return  $V^{\mathcal{G}_i}$ 

procedure McN-ZOdd( $\mathcal{G}, d$ ):
   $i \leftarrow 0$ ;  $\mathcal{G}_1 \leftarrow \mathcal{G}$ 
  repeat
     $i \leftarrow i + 1$ 
     $D_i \leftarrow \pi^{-1}(d) \cap \mathcal{G}_i$ 
     $\mathcal{G}'_i \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(D_i)$ 
     $U_i \leftarrow \text{McN-Z}_{\text{Even}}(\mathcal{G}'_i, d - 1)$ 
     $\mathcal{G}_{i+1} \leftarrow \mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(U_i)$ 
  until  $U_i = \emptyset$ 
   $\perp$  return  $V^{\mathcal{G}_i}$ 

```

ALGORITHM 2: *McNaughton-Zielonka algorithm*

The classic recursive McNaughton-Zielonka algorithm (Algorithm 2) computes the largest dominia in a parity game. In order to obtain the largest Even dominion in a parity game \mathcal{G} , it suffices to call $\text{McN-Z}_{\text{Even}}(\mathcal{G}, d)$, where d is even and all vertex priorities in \mathcal{G} are at most d . In order to obtain the largest Odd dominion in a parity game \mathcal{G} , it suffices to call $\text{McN-Z}_{\text{Odd}}(\mathcal{G}, d)$, where d is odd and all vertex priorities in \mathcal{G} are at most d .

The procedures $\text{McN-Z}_{\text{Even}}$ and $\text{McN-Z}_{\text{Odd}}$ are mutually recursive and whenever a recursive call is made, the second argument d decreases by 1. Figure 2 illustrates one iteration of the main loop in a call of procedure $\text{McN-Z}_{\text{Even}}$. The outer rectangle denotes subgame \mathcal{G}_i , the thin horizontal rectangle at the top denotes the set D_i of the vertices in \mathcal{G}_i whose priority is d , and the set below the horizontal wavy line is subgame \mathcal{G}'_i , which is the set of vertices in \mathcal{G}_i that are not in the attractor $\text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$. The recursive call of $\text{McN-Z}_{\text{Odd}}$ returns the set U_i , and \mathcal{G}_{i+1} is the subgame to the left of the vertical zig-zag line, and it is induced by the set of vertices in \mathcal{G}_i that are not in the attractor $\text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(U_i)$.

A way to prove the correctness of McNaughton-Zielonka algorithm we wish to highlight here is to enhance the algorithm slightly to produce not just a set of vertices but also an Even attractor decomposition of the set and an Odd attractor decomposition of its complement. We explain how to modify procedure $\text{McN-Z}_{\text{Even}}$ and leave it as an exercise for the reader to analogously modify procedure $\text{McN-Z}_{\text{Odd}}$. In procedure $\text{McN-Z}_{\text{Even}}(\mathcal{G}, d)$, replace the line

$$U_i \leftarrow \text{McN-Z}_{\text{Odd}}(\mathcal{G}'_i, d - 1)$$

by the line

$$U_i, \mathcal{H}_i, \mathcal{H}'_i \leftarrow \text{McN-Z}_{\text{Odd}}(\mathcal{G}'_i, d-1).$$

Moreover, if upon termination of the **repeat-until** loop we have

$$\mathcal{H}_i = \langle \emptyset, (S_1, \mathcal{I}_1, A_1), \dots, (S_k, \mathcal{I}_k, A_k) \rangle$$

then instead of returning just the set $V^{\mathcal{G}_i}$, let the procedure return both $V^{\mathcal{G}_i}$ and the following two objects:

$$\langle \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i), (S_1, \mathcal{I}_1, A_1), \dots, (S_k, \mathcal{I}_k, A_k) \rangle \quad (1)$$

and

$$\langle \emptyset, (U_1, \mathcal{H}'_1, \text{Attr}_{\text{Odd}}^{\mathcal{G}_1}(U_1)), \dots, (U_i, \mathcal{H}'_i, \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(U_i)) \rangle \quad (2)$$

In an inductive argument by induction on d and i , the inductive hypothesis is that:

- \mathcal{H}'_i is an Odd $(d-1)$ -attractor decomposition of the subgame $\mathcal{G}'_i \cap U_i$;
- \mathcal{H}_i is an Even d -attractor decomposition of the subgame $\mathcal{G}'_i \setminus U_i$;

and the inductive step is then to show that:

- for every i , (2) is an Odd $(d+1)$ -attractor decomposition of subgame $\mathcal{G} \setminus \mathcal{G}_{i+1}$;
- upon termination of the **repeat-until** loop, (1) is an Even d -attractor decomposition of subgame \mathcal{G}_{i+1} .

The general arguments in such a proof are well known [McN93, Zie98, JPZ08, DJL18] and hence we omit the details here.

B. EMBEDDABLE DECOMPOSITION THEOREM

Theorem 14 (Embeddable decomposition). *If T is a trap for Even in a parity game \mathcal{G} and $\mathcal{G}' = \mathcal{G} \cap T$ is the subgame induced by T , then for every Even attractor decomposition \mathcal{H} of \mathcal{G} , there is an Even attractor decomposition \mathcal{H}' of \mathcal{G}' , such that $\mathcal{T}_{\mathcal{H}}$ embeds $\mathcal{T}_{\mathcal{H}'}$.*

First stated at page 10.

Proof of Theorem 14. Without loss of generality, assume that d is even and

$$\mathcal{H} = \langle A, (S_1, \mathcal{H}_1, A_1), \dots, (S_k, \mathcal{H}_k, A_k) \rangle$$

is an Even d -attractor decomposition of \mathcal{G} , where A is the Even attractor to the set D of vertices of priority d in \mathcal{G} . In Figure 3, set T and the subgame \mathcal{G}' it induces form the pentagon obtained from the largest rectangle by removing the triangle above the diagonal line in the top-left corner. Sets A , S_1 , and A_1 are also illustrated, together with sets A' , S'_1 , A'_1 and subgames \mathcal{G}_1 , \mathcal{G}_2 , \mathcal{G}'_1 , and \mathcal{G}'_2 , which are defined as follows.

Let $\mathcal{G}_1 = \mathcal{G} \setminus A$, and $\mathcal{G}_2 = \mathcal{G}_1 \setminus A_1$. We will define sets A' , S'_1 , A'_1 , \dots , S'_ℓ , A'_ℓ , and Even $(d-2)$ -attractor decompositions $\mathcal{H}'_1, \dots, \mathcal{H}'_\ell$ of subgames $\mathcal{G} \cap S'_1, \dots, \mathcal{G} \cap S'_\ell$, respectively, such that

$$\mathcal{H}' = \langle A', (S'_1, \mathcal{H}'_1, A'_1), \dots, (S'_\ell, \mathcal{H}'_\ell, A'_\ell) \rangle$$

is an Even d -attractor decomposition of subgame \mathcal{G}' and $\mathcal{T}_{\mathcal{H}}$ embeds $\mathcal{T}_{\mathcal{H}'}$.

Let A' be the Even attractor to $D \cap T$ in \mathcal{G}' and let $\mathcal{G}'_1 = \mathcal{G}' \setminus A'$. Set $S'_1 = S_1 \cap \mathcal{G}'_1$, let A'_1 be the Even attractor to S'_1 in \mathcal{G}'_1 , and let $\mathcal{G}'_2 = \mathcal{G}'_1 \setminus A'_1$.

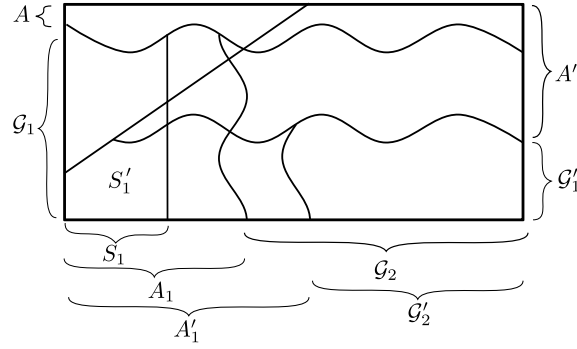


FIGURE 3: *Attractors, subgames, and dominia in the proof of the embeddable decomposition theorem.*

Firstly, since $D \subseteq V^G$ and T is a trap for Even in G , by Proposition 16, we have that G'_1 is a trap for Even in subgame G_1 . Since $S_1 \subseteq V^{G_1}$ and subgame G'_1 is a trap for Even in subgame G_1 , again by Proposition 16, we conclude that G'_2 is a trap for Even in subgame G_2 .

Secondly, we argue that S'_1 is an Even dominion in subgame G'_1 . This follows by recalling that S_1 is a dominion for Even in G_1 and G'_1 is a trap for Even in G_1 , and then applying Proposition 15.

Thirdly, we argue that S'_1 is a trap for Even in subgame $G \cap S_1$. This follows by recalling that S_1 is a trap for Odd in G_1 and that G'_1 is a trap for Even in G_1 , and then applying Proposition 15.

We are now in a position to apply the inductive hypothesis twice in order to complete the definition of the attractor decomposition \mathcal{H}' . Firstly, recall that S'_1 is a trap for Even in subgame $G \cap S_1$ and that \mathcal{H}_1 is a $(d-2)$ -attractor decomposition of $G \cap S_1$, so we can apply the inductive hypothesis to obtain a $(d-2)$ -attractor decomposition \mathcal{H}'_1 of subgame $G \cap S_1$, such that $\mathcal{T}_{\mathcal{H}_1}$ embeds $\mathcal{T}_{\mathcal{H}'_1}$. Secondly, note that

$$\mathcal{J} = \langle \emptyset, (S_2, \mathcal{H}_2, A_2), \dots, (S_k, \mathcal{H}_k, A_k) \rangle$$

is a d -attractor decomposition of G_2 . We find a d -attractor decomposition \mathcal{J}' of subgame G'_2 , such that $\mathcal{T}_{\mathcal{J}}$ embeds $\mathcal{T}_{\mathcal{J}'}$. Recalling that G'_2 is a trap for Even in subgame G_2 , it suffices to use the inductive hypothesis for subgame G'_2 of game G_2 and the d -attractor decomposition \mathcal{J} of G_2 .

Verifying that \mathcal{H}' is a d -attractor decomposition of G' is routine. That $\mathcal{T}_{\mathcal{H}}$ embeds $\mathcal{T}_{\mathcal{H}'}$ also follows routinely from $\mathcal{T}_{\mathcal{H}_1}$ embedding $\mathcal{T}_{\mathcal{H}'_1}$ and $\mathcal{T}_{\mathcal{J}}$ embedding $\mathcal{T}_{\mathcal{J}'}$. \square

C. DOMINION SEPARATION THEOREM

Before we prove the dominion separation theorem: we recall a simple proposition from Lehtinen, Parys, Schewe and Wojtczak [LPSW22]. Note that it is a straightforward corollary of the dual of Proposition 16 (in case $B \cap T = \emptyset$).

Proposition 24. *If T is a trap for Odd in G and $T \cap B = \emptyset$ then we also have that $T \cap \text{Attr}_{\text{Odd}}^G(B) = \emptyset$.*

Theorem 17 (Dominion separation). *Let G be an (n, d) -small parity game and let $\mathcal{T}^{\text{Even}} = \langle \mathcal{T}_1^{\text{Even}}, \dots, \mathcal{T}_\ell^{\text{Even}} \rangle$ and $\mathcal{T}^{\text{Odd}} = \langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_k^{\text{Odd}} \rangle$ be trees of height at most $\lceil d/2 \rceil$ and $\lfloor d/2 \rfloor$,*

First stated at page 11.

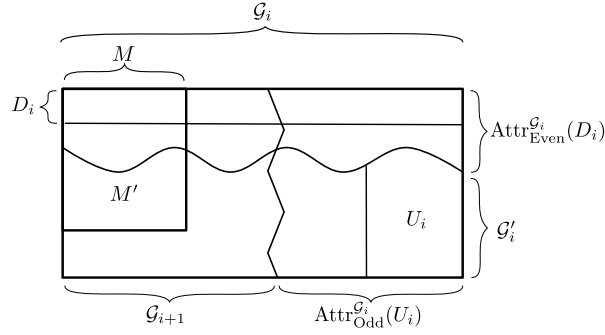


FIGURE 4: Attractors, subgames, and dominia in the first part of the proof of the dominion separation theorem.

respectively.

- (a) If d is even and $\mathcal{G}_1, \dots, \mathcal{G}_{k+1}$ are the games that are computed in the successive iterations of the loop in the call $\text{UnivEven}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$, then for every $i = 0, 1, \dots, k$, we have that \mathcal{G}_{i+1} separates every Even *dominion* in \mathcal{G} that tree $\mathcal{T}^{\text{Even}}$ embeds from every Odd *dominion* in \mathcal{G} that tree $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_i^{\text{Odd}} \rangle$ embeds.
- (b) If d is odd and $\mathcal{G}_1, \dots, \mathcal{G}_{\ell+1}$ are the games that are computed in the successive iterations of the loop in the call $\text{UnivOdd}(\mathcal{G}, d, \mathcal{T}^{\text{Even}}, \mathcal{T}^{\text{Odd}})$, then for every $i = 0, 1, \dots, \ell$, we have that \mathcal{G}_{i+1} separates every Odd *dominion* in \mathcal{G} that tree \mathcal{T}^{Odd} embeds from every Even *dominion* in \mathcal{G} that tree $\langle \mathcal{T}_1^{\text{Even}}, \dots, \mathcal{T}_i^{\text{Even}} \rangle$ embeds.

Proof of Theorem 17. We prove the statement of part (a); the proof of part (b) is analogous.

The proof is by induction on the height of tree $\mathcal{T}^{\text{Odd}} \bowtie \mathcal{T}^{\text{Even}}$ (the “outer” induction). If the height is 0 then tree \mathcal{T}^{Odd} is the trivial tree $\langle \rangle$; hence $k = 0$, the algorithm returns the set $V^{\mathcal{G}_1} = V^{\mathcal{G}}$, which contains the largest Even dominion, and which is trivially disjoint from the largest Odd dominion (because the latter is empty).

If the height of $\mathcal{T}^{\text{Odd}} \bowtie \mathcal{T}^{\text{Even}}$ is positive, then we split the proof of the separation property into two parts.

Even dominia embedded by $\mathcal{T}^{\text{Even}}$ are included in \mathcal{G}_{i+1} . We prove by induction on i (the “inner” induction) that for $i = 0, 1, 2, \dots, k$, if M is an Even dominion in \mathcal{G} that $\mathcal{T}^{\text{Even}}$ embeds, then $M \subseteq \mathcal{G}_{i+1}$.

For $i = 0$, this is moot because $\mathcal{G}_1 = \mathcal{G}$.

For $i > 0$, let M be an Even dominion that has an Even d -attractor decomposition \mathcal{H} such that $\mathcal{T}^{\text{Even}}$ embeds $\mathcal{T}_{\mathcal{H}}$. The inner inductive hypothesis (for $i - 1$) implies that $M \subseteq \mathcal{G}_i$.

The reader is encouraged to systematically refer to Figure 4 to better follow the rest of this part of the proof.

Let $M' = M \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$. Because $\mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$ is a trap for Even in \mathcal{G}_i and M is a trap for Odd in \mathcal{G}_i , the dual of Proposition 15 yields that M' is a trap for Even in $\mathcal{G}_i \cap M$.

Then, because \mathcal{H} is an Even d -attractor decomposition of $\mathcal{G} \cap M$, it follows by Theorem 14 that there is an Even d -attractor decomposition \mathcal{H}' of $\mathcal{G}_i \cap M'$ such that $\mathcal{T}_{\mathcal{H}}$ embeds $\mathcal{T}_{\mathcal{H}'}$, and hence also $\mathcal{T}^{\text{Even}}$ embeds $\mathcal{T}_{\mathcal{H}'}$.

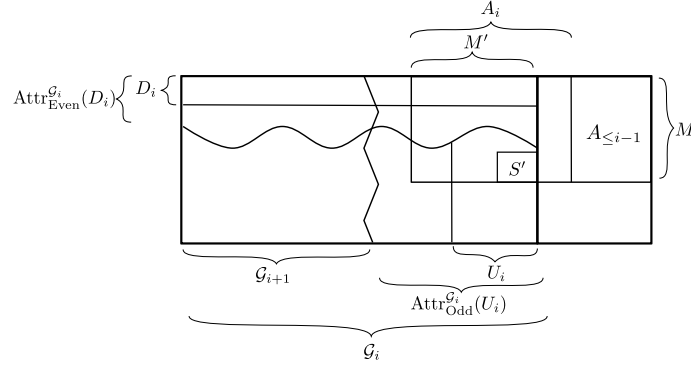


FIGURE 5: *Attractors, subgames, and dominia in the second part of the proof of the dominion separation theorem.*

Therefore, because M' is an Even dominion in the game $\mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$, part (b) of the outer inductive hypothesis yields $M' \cap U_i = \emptyset$.

Finally, because $M \setminus M' \subseteq \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$ and $(M' \setminus M) \cap U_i = \emptyset$, it follows that $M \cap U_i = \emptyset$. By Proposition 24, we obtain $M \cap \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(U_i) = \emptyset$ and hence $M \subseteq \mathcal{G}_{i+1}$.

Odd dominia embedded by $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_i^{\text{Odd}} \rangle$ are disjoint from \mathcal{G}_{i+1} . We prove by induction on i (another “inner” induction) that for $i = 0, 1, \dots, k$, if M is an Odd dominion in \mathcal{G} that $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_i^{\text{Odd}} \rangle$ embeds, then $\mathcal{G}_{i+1} \cap M = \emptyset$.

For $i = 0$, note that $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_i^{\text{Odd}} \rangle = \langle \rangle$ and the only Odd dominion M in \mathcal{G} that has an Odd $(d+1)$ -attractor decomposition whose tree is the trivial tree $\langle \rangle$ is the empty set, and hence $\mathcal{G}_1 \cap M = \emptyset$, because $\mathcal{G}_1 = \mathcal{G}$.

The reader is encouraged to systematically refer to Figure 5 to better follow the rest of this part of the proof.

For $i > 0$, let

$$\mathcal{H} = \langle \emptyset, (S_1, \mathcal{H}_1, A_1), \dots, (S_{\bar{i}}, \mathcal{H}_{\bar{i}}, A_{\bar{i}}) \rangle$$

be an Odd $(d+1)$ -attractor decomposition of $\mathcal{G} \cap M$ such that $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_i^{\text{Odd}} \rangle$ embeds $\mathcal{T}_{\mathcal{H}}$. Note that the embedding implies that $\bar{i} \leq i$.

If $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_{\bar{i}-1}^{\text{Odd}} \rangle$ embeds $\mathcal{T}_{\mathcal{H}}$ then the inner inductive hypothesis (for $i-1$) implies that $\mathcal{G}_i \cap M = \emptyset$ and thus $\mathcal{G}_{i+1} \cap M = \emptyset$ since $\mathcal{G}_{i+1} \subseteq \mathcal{G}_i$.

Otherwise, it must be the case that

$$\mathcal{T}_i^{\text{Odd}} \text{ embeds } \mathcal{T}_{\mathcal{H}_{\bar{i}}}. \quad (3)$$

Observe that the set $A_{\leq \bar{i}-1} = A_1 \cup A_2 \cup \dots \cup A_{\bar{i}-1}$ is a trap for Even in $\mathcal{G} \cap M$, and hence by trap transitivity it is a trap for Even in \mathcal{G} because M is a trap for Even in \mathcal{G} . Moreover, subgame $\mathcal{G} \cap A_{\leq \bar{i}-1}$ has an Odd $(d+1)$ -attractor decomposition

$$\mathcal{J} = \langle \emptyset, (S_1, \mathcal{H}_1, A_1), \dots, (S_{\bar{i}-1}, \mathcal{H}_{\bar{i}-1}, A_{\bar{i}-1}) \rangle$$

in \mathcal{G} and hence—by the dual of Proposition 4—it is an Odd dominion in \mathcal{G} , and ordered tree $\langle \mathcal{T}_1^{\text{Odd}}, \dots, \mathcal{T}_{\bar{i}-1}^{\text{Odd}} \rangle$ embeds $\mathcal{T}_{\mathcal{J}}$. Hence, the inner inductive hypothesis (for $i-1$) yields

$$\mathcal{G}_i \cap A_{\leq \bar{i}-1} = \emptyset. \quad (4)$$

Set $M' = \mathcal{G}_i \cap M$ and note that not only $M' \subseteq A_i$, but also M' is a trap for Odd in A_i , because \mathcal{G}_i is a trap for Odd in \mathcal{G} . Moreover—by Proposition 15— M' is an Odd dominion in \mathcal{G}_i because \mathcal{G}_i is a trap for Odd in \mathcal{G} and M is a dominion for Odd in \mathcal{G} .

Observe that $\mathcal{J} = \langle \emptyset, (S_i, \mathcal{H}_i, A_i) \rangle$ is an Odd $(d+1)$ -attractor decomposition of $\mathcal{G} \cap A_i$. By the embeddable decomposition theorem (Theorem 14), it follows that there is an Odd $(d+1)$ -attractor decomposition \mathcal{K} of $\mathcal{G} \cap M'$ such that $\mathcal{T}_{\mathcal{J}}$ embeds $\mathcal{T}_{\mathcal{K}}$. Because of this embedding, \mathcal{K} must have the form $\mathcal{K} = \langle \emptyset, (S', \mathcal{K}', M') \rangle$. Since $\mathcal{T}_{\mathcal{J}}$ embeds $\mathcal{T}_{\mathcal{K}}$, we also have that $\mathcal{T}_{\mathcal{H}_i}$ embeds $\mathcal{T}_{\mathcal{K}'}$, and hence—by (3)— $\mathcal{T}_i^{\text{Odd}}$ embeds $\mathcal{T}_{\mathcal{K}'}$.

Note that S' is a trap for Odd in $\mathcal{G} \cap M'$ in which every vertex priority is at most $d-1$, because \mathcal{K} is an Odd $(d+1)$ -attractor decomposition of $\mathcal{G} \cap M'$. It follows that S' is also an Odd dominion in $\mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$.

The outer inductive hypothesis then yields $S' \subseteq U_i$. It follows that

$$M' = \text{Attr}_{\text{Odd}}^{\mathcal{G}_i \cap M'}(S') \subseteq \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(S') \subseteq \text{Attr}_{\text{Odd}}^{\mathcal{G}_i}(U_i),$$

where the first inclusion holds because M' is a trap for Even in \mathcal{G}_i , and the second follows from monotonicity of the attractor operator. When combined with (4), this implies $\mathcal{G}_{i+1} \cap M = \emptyset$. \square

D. FLOWERINESS LEMMA

In this Appendix, we will prove Lemma 19 and Theorem 20. Whenever we want to denote the fixpoint obtained by repeated application of a monotone function f on a set, we call this f^* . Before we embark on the proofs, we would like to call attention to following property of flowery subgames. It shows how complements of two specific kinds of flowery sets result in another flowery subgame. We will use this property in several of our proofs.

Property 25. For $A \subseteq Y \subseteq X \subseteq (U \times [d])$, we have:

$$\mathcal{F}(X, Y) \setminus \mathcal{F}(X, A) = \mathcal{F}(X \setminus A, Y \setminus A).$$

Notice that $\mathcal{F}(X \setminus A, Y \setminus A) = \mathcal{F}(Z \cup W, W)$, where $Z = X \setminus Y$ and $W = X \setminus A$.

Consider the following proposition useful in the proof of the Lemma 19.

Proposition 26. Given a fixpoint game \mathcal{G}_f , after removing the Even attractor to the set of Odd vertices with no outgoing edges and the Odd attractor to the Even vertices with no outgoing edges, we are left with a flowery subgame.

Proof. The game \mathcal{G}_f contains exactly the vertices in the subgame $\mathcal{F}(U \times [d], U \times [d])$ along with v_{\emptyset} .

- Initially, we remove the only Odd vertex with no outgoing edge: v_{\emptyset} , along with its Even attractor. The Even attractor to v_{\emptyset} in \mathcal{G}_f is exactly all the vertices of the flowery set $\mathcal{F}(Z, Z)$ and v_{\emptyset} , where $Z = f^*(\emptyset)$ winning for Even. The remaining subgame after removing these vertices is the flowery subgame $\mathcal{F}(U \times [d], (U \times [d]) \setminus Z)$ from Property 25.
- Let us call the flowery subgame obtained from the above procedure $\mathcal{F}(X, Y)$. Observe that if $Y \subseteq f(X)$, then there is always an outgoing edge for each vertex in the subgame. If not, we remove the Odd attractor to the set of Even vertices with no outgoing edges: $Y \cap \bar{f}(X)$. The complement of this Odd attractor turns out to be the flowery subgame $\mathcal{F}(X, Y \setminus \bar{f}(X))$ from Property 25. \square

Assuming now that we always have outgoing edges in flowery subgames, we consider the following Lemma which shows how we can compute attractors to sets in these subgames with at most $d \cdot |U|$ many calls to the function f .

Let us now prove Lemma 19 by instead proving a stronger statement stated in Lemma 31. To lead to the proof of Lemma 31, we need Lemma 27 which states intuitively that computing attractors to specific flowery subgames lead to specific flowery subgames whose complement is also flowery.

Lemma 27. *In a flowery subgame $\mathcal{G} = \mathcal{F}(X, Y)$:*

- (a) *the Even attractor to a set of Even vertices $A \subseteq Y$ in $\mathcal{G} = \mathcal{F}(X, Y)$ where $Z = X \setminus Y$ is*

$$\mathcal{F}(Z \cup \text{Pre}_{\mathcal{G}, \text{Even}}^*(A), \text{Pre}_{\mathcal{G}, \text{Even}}^*(A))$$

where $\text{Pre}_{\mathcal{G}, \text{Even}}(A) = (f(Z \cup A) \cap Y) \cup A$;

- (b) *the Odd attractor to a set of Even vertices A or a subgame $\mathcal{F}(X, A)$ in $\mathcal{G} = \mathcal{F}(X, Y)$ is*

$$\mathcal{F}(X, \text{Pre}_{\mathcal{G}, \text{Odd}}^*(A))$$

where $\text{Pre}_{\mathcal{G}, \text{Odd}}(A) = \overline{(f(X \setminus A) \cap Y)} \cup A$.

We will break down our Lemma into Propositions 28 and 29 which will result in Corollary 30 from which Lemma 27 follows.

Proposition 28. *In a flowery subgame $\mathcal{G} = \mathcal{F}(X, Y)$ and $A \subseteq Y$, the flowery subgame $\mathcal{F}(Z \cup \text{Pre}_{\mathcal{G}, \text{Even}}(A), \text{Pre}_{\mathcal{G}, \text{Even}}(A))$ is exactly the set of vertices from which Even has a strategy to visit A in at most three steps, where $\text{Pre}_{\mathcal{G}, \text{Even}}(A) = (f(Z \cup A) \cap Y) \cup A$.*

Proof. We will argue about vertices from which Even has a strategy to visit vertices in A in at most one, two and three steps below.

- (1) Consider any Odd vertex v_B where $B \subseteq Z \cup A$ and the intersection of B with A is non empty. From such a v_B , in one step, Even can ensure that a play reaches A . All such vertices v_B along with the core A is exactly denoted by the vertices of the subgame $\mathcal{F}(Z \cup A, A)$.
- (2) We will show that from any Even vertex $(u, i) \in \text{Pre}_{\mathcal{G}, \text{Even}}(A) = f(Z \cup A) \cap Y \cup A$, there is a strategy for Even to reach a Even vertex in A in at most two steps. To show this, we will show that:
 - (\Rightarrow) in one step, Even can move to some Odd vertex $v_B \in \mathcal{F}(Z \cup A, A)$;
 - (\Leftarrow) from vertices not in $\text{Pre}_{\mathcal{G}, \text{Even}}(A)$, all of Even's outgoing edges lead to a vertex not in $\mathcal{F}(Z \cup A, A)$.

To show the forward direction, let $(u, i) \in (f(Z \cup A) \cap Y) \cup A$, if $(u, i) \in A$ then we are done, if not, the strategy for Even from (u, i) is to choose the Odd vertex $v_{Z \cup A}$ and such an edge exists since $(u, i) \in f(Z \cup A)$, and this Odd vertex is in the flowery subgame $\mathcal{F}(Z \cup A, A)$.

To show the reverse direction, Consider $(u, i) \notin f(Z \cup A) \cup A$ but $(u, i) \in Y$. All edges out of the Even vertex (u, i) leads to an Odd vertex v_B in $\mathcal{F}(X, Y)$ such that B has some element other than from Z or A i.e, $B \setminus (Z \cup A) \neq \emptyset$. This follows from the monotonicity of f along with our assumption that $(u, i) \notin f(Z \cup A)$. After one step, the game is at an Odd vertex v_B that it is not in $\mathcal{F}(Z \cup A, A)$.

- (3) The argument to conclude that $\mathcal{F}(X, \text{Pre}_{\mathcal{G}, \text{Odd}}(A))$ is exactly the set we desire is similar to (1). □

Proposition 29. *In a flowery subgame $\mathcal{G} = \mathcal{F}(X, Y)$ and $A \subseteq Y$, From any vertex of the flowery subgame $\mathcal{F}(X, \text{Pre}_{\mathcal{G}, \text{Odd}}(A))$, Odd has a strategy to visit a set of Even vertices A in at most three steps where $\text{Pre}_{\mathcal{G}, \text{Odd}}(A) = (\overline{f(X \setminus A)} \cap Y) \cup A$. The above subgame is the exact set of vertices from which Odd has such a strategy.*

Proof. We show the set of vertices from which Even has a strategy to visit vertices in A in at most one, two and three steps below.

- (1) From vertex v_B where B of X which intersects with A non-trivially, Odd would be able to reach a vertex in A in at most one step. This exactly is all the Odd vertices in the flowery subgame $\mathcal{F}(X, A)$.
- (2) We will show that in one step, Odd has a strategy to visit the subgame $\mathcal{F}(X, A)$ from vertices in $\text{Pre}_{\mathcal{G}, \text{Odd}}(A) \cup A$. We do this by showing inclusion in two direction.
 - (\Rightarrow) Consider $(v, j) \in \text{Pre}_{\mathcal{G}, \text{Odd}}(A) = (\overline{f(X \setminus A)} \cap Y) \cup A$. If $(v, j) \notin A$, then $(v, j) \in Y$ and $\overline{f(X \setminus A)}$. Mainly note that $(v, j) \notin f(X \setminus A)$. Since all subgames are such that there is always an outgoing edge and given that f is monotone, any Odd vertex v_B in $\mathcal{F}(X, Y)$ which has an edge to it from (v, j) must be such that $B \cap A \neq \emptyset$. For any choice successors from (v, j) of Even will lead to a vertex B which intersects with A and hence there is a strategy for Odd to move to a vertex in (u, i) in $B \cap A$.
 - (\Leftarrow) Now we need to show a strategy for Even to remain in the complement of the game $\mathcal{F}(X, \text{Pre}_{\mathcal{G}, \text{Odd}}(A))$ for two steps from all other Even vertices. Let us denote $\text{Pre}_{\mathcal{G}, \text{Odd}}(A)$ by W . Note that the complement of $\mathcal{F}(X, W)$ in $\mathcal{F}(X, Y)$ is $\mathcal{F}(X \setminus W, Y \setminus W)$. Notice that

$$Y \setminus W = Y \setminus (\overline{f(X \setminus A)} \cup A)$$

So, any $(w, j) \in Y \setminus Z$ is in Y and since $(w, j) \notin W$, $(w, j) \in f(X \setminus A)$. This means that from any such (w, j) , Even can choose the vertex v_B in $\mathcal{F}(X \setminus W, Y \setminus W)$ where $B \subseteq X \setminus A$, making sure that in the next step Odd will not be able to take the play to an Even vertex in A .

- (3) From the structure of the game, it is easy to see that any v_B such that B intersects with $\text{Pre}_{\mathcal{G}, \text{Odd}}(A) \cup A$ would be able to visit an element in $\text{Pre}_{\mathcal{G}, \text{Odd}}(A) \cup A$, which we have shown is exactly the set of vertices from which Odd could force the play in at most two steps to visit A . \square

From the proof of the Propositions 28 and 29, we can extend these to show the following Corollary from which Lemma 27 follows.

Corollary 30. *In a flowery subgame $\mathcal{G} = \mathcal{F}(X, Y)$ and $A \subseteq Y$,*

- *The flowery subgame $\mathcal{F}(Z \cup \text{Pre}_{\mathcal{G}, \text{Even}}(A), \text{Pre}_{\mathcal{G}, \text{Even}}(A))$ is the set of vertices from which Even has a strategy to visit the vertices in $\mathcal{F}(Z \cup A, A)$ in at most two steps, where $\text{Pre}_{\mathcal{G}, \text{Even}}(A) = (f(Z \cup A) \cap Y) \cup A$;*
- *The vertices of $\mathcal{F}(X, \text{Pre}_{\mathcal{G}, \text{Odd}}(A))$ is the set of vertices from which Odd has a strategy to visit a vertex in $\mathcal{F}(X, A)$ in at most two steps.*

We state that Lemma 27 follows naturally from Corollary 30 and conclude the proof of Lemma 27.

We will now proceed to the main proof of the section:

Lemma 19 (Floweriness). *If $\text{Univ}_{\text{Even}}$ (resp., Univ_{Odd}) is run on a flowery subgame, for all iterations in the for-loop, subgame \mathcal{G}_i is also flowery. In particular, \mathcal{G}_{k+1} , which is the subgame returned, is flowery.*

First stated at page 14.

We will instead prove a stronger version of Lemma 19, stated below:

- Lemma 31.** (i) If $\text{Univ}_{\text{Even}}$ is run on a flowery subgame $\mathcal{F}(X, Y)$, then in all iterations in the for-loop in the subgame \mathcal{G}_i is of the form $\mathcal{F}(X \setminus A'_i, Y \setminus A'_i)$ for $A'_i \subseteq Y$, in particular \mathcal{G}_{k+1} , which is the set of vertices returned.
- (ii) If Univ_{Odd} is run on a flowery subgame $\mathcal{F}(X, Y)$, then in all iterations in the for-loop, the subgame \mathcal{G}_i is of the form $\mathcal{F}(X, Y \setminus A'_i)$, where $A'_i \subseteq Y$ in particular \mathcal{G}_{k+1} , which is the set of vertices returned.

Proof of Lemmas 19 and 31. We will prove this by induction on the sum of the number of vertices in these subgames and the number of vertices on which these calls are made. For the base case, with an empty set irrespective of any priority, the above statement is trivially true. We will now prove that (i) and (ii) hold for games with at least one vertex and trees $\mathcal{T}_{\text{Even}}$ and \mathcal{T}_{Odd} . The proof follows from Lemma 27 and induction as shown.

(i) Since $\mathcal{G}_1 = \mathcal{G} = \mathcal{F}(X, Y)$, we show that if \mathcal{G}_i is of the form $\mathcal{F}(X \setminus A'_i, Y \setminus A'_i)$ where $A_i \subseteq Y$. For convenience, we will call $X \setminus A'_i$ as X_i and $Y \setminus A'_i$ as Y_i . We will show that \mathcal{G}_{i+1} is of the form $\mathcal{F}(X \setminus A'_{i+1}, Y \setminus A'_{i+1})$ by showing that in fact it is $\mathcal{F}(X_i \setminus A'_{i+1}, Y_i \setminus A'_{i+1})$ for some $A'_{i+1} \subseteq Y$. First notice that $\mathcal{G}'_i = \mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i)$, where D_i is some subset of Even vertices. From Lemma 28, we have for $Z = X \setminus Y$,

$$\mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i) = \mathcal{F}(X_i, Y'_i) \setminus \mathcal{F}(Z \cup \text{Pre}_{\mathcal{G}_i, \text{Even}}^*(D_i), \text{Pre}_{\mathcal{G}_i, \text{Even}}^*(D_i))$$

Since $Z = X \setminus Y = X_i \setminus Y_i$, we have

$$\mathcal{G}'_i = \mathcal{G}_i \setminus \text{Attr}_{\text{Even}}^{\mathcal{G}_i}(D_i) = \mathcal{F}(X_i, Y_i \setminus \text{Pre}_{\mathcal{G}_i, \text{Even}}^*(D_i))$$

The U_i computed by performing Univ_{Odd} on \mathcal{G}'_i must be of the form $\mathcal{F}(X_i, Z_i)$ for $Z_i \subseteq Y_i$ by induction and the attractor to U_i , must be of the form $\mathcal{F}(X_i, W_i)$ from Proposition 28. Hence

$$\mathcal{G}_{i+1} = \mathcal{F}(X_i, Y_i) \setminus \mathcal{F}(X_i, W_i) = \mathcal{F}(X_i \setminus W_i, X_i \setminus W_i).$$

(ii) We will show that if \mathcal{G}_i is of the form $\mathcal{F}(X, Y_i)$, then \mathcal{G}_{i+1} is of the form $\mathcal{F}(X, Y_{i+1})$ for $Y_{i+1} \subseteq Y_i$. In each iteration i , the Odd attractor to D_i in \mathcal{G}_i is of the form $\mathcal{F}(X, A_i)$. This shows that \mathcal{G}'_i , which is obtained by removing the Odd attractor $\mathcal{F}(X, A_i)$ from \mathcal{G}_i is of the form $\mathcal{F}(X_i \setminus A_i, Y_i \setminus A_i)$. Running Univ_{Odd} on \mathcal{G}_i gives U_i of the form $\mathcal{F}(X_i \setminus W_i, Y_i \setminus W_i)$ by induction, and an Even attractor to the set $\mathcal{F}(X_i \setminus W_i, Y_i \setminus W_i)$ would be of the flowery subgame $\mathcal{F}(X_i \setminus W'_i, Y_i \setminus W'_i)$ for some $W'_i \subseteq W_i$. So, \mathcal{G}_{i+1} , which is obtained from removing this Even attractor from \mathcal{G}_i would be obtained as follows

$$\mathcal{G}_{i+1} = \mathcal{G}_i \setminus \mathcal{F}(X_i \setminus W'_i, Y_i \setminus W'_i) = \mathcal{F}(X, Y_i \setminus W'_i). \quad \square$$

E. SYMBOLIC ALGORITHM

In this appendix we describe how the number of symbolic set variables in the symbolic implementation of the universal algorithm can be further reduced from $O(d)$ to $O(\log d)$, leading to Theorem 23.

Theorem 23. *There exists a symbolic algorithm that solves (n, d) -small parity games using $O(\lg d)$ symbolic set variables, $O(\log d \cdot \log n)$ bits of conventional space, and whose running time is polynomial if $d = O(\log n)$, and quasi-polynomial, namely $n^{2 \lg(d/\lg n) + O(1)}$, if $d = \omega(\log n)$.*

First stated at page 16.

Proof of Theorem 23. We use letters G , D , G' , and U to denote the sets V^{G_i} , D_i , $V^{G'_i}$, and U_i for some i -th iteration of any of the recursive calls of the universal algorithm. Observe that we do not need to keep the symbolic variables that store the sets D , G' , and U on the stack of recursive calls because on any return from a recursive call, their values are not needed to proceed. How can we store the sets denoted by all the symbolic set variables G on the stack using only $O(\log d)$ symbolic set variables, while the height of the stack may be as large as d ?

Firstly, we argue that we can symbolically represent a sequence $\langle G_{d-1}, \dots, G_i \rangle$ of set variables that would normally occur on the stack of recursive calls of the universal algorithm, by another sequence $\langle H_{d-1}, \dots, H_0 \rangle$, in which the sets form a partition of the set of vertices in the parity game. Indeed, a sequence $\langle G_d, \dots, G_i \rangle$ on the stack of recursive calls at any time forms a descending chain w.r.t. inclusion, and G_d is the set of all vertices, so it suffices to consider the sequence $\langle G_d \setminus G_{d-1}, \dots, G_{i+1} \setminus G_i, G_i, \emptyset, \dots, \emptyset \rangle$.

Secondly, we argue that the above family of d mutually disjoint sets can be succinctly represented and maintained using $O(\log d)$ set variables. W.l.o.g., assume that d is a power of 2. For every $k = 1, 2, \dots, \lg d$, and for every $i = 1, 2, \dots, d$, let $\text{bit}_k(i)$ be the k -th digit in the binary representation of i (and zero if there are less than k digits). We now define the following sequence of sets $\langle S_1, S_2, \dots, S_{\lg d} \rangle$ that provides a succinct representation of the sequence $\langle H_{d-1}, \dots, H_0 \rangle$. For every $k = 1, 2, \dots, \lg d$, we set:

$$S_k = \bigcup \{H_i : 0 \leq i \leq d-1 \text{ and } \text{bit}_k(i) = 1\}.$$

By sets $\langle H_{d-1}, \dots, H_0 \rangle$ forming a partition of the set of all vertices, it follows that for every $i = 0, 1, \dots, d-1$, we have:

$$H_i = \bigcap \{S_k : 1 \leq k \leq \lg d \text{ and } \text{bit}_k(i) = 1\} \cap \bigcap \{\bar{S}_k : 1 \leq k \leq \lg d \text{ and } \text{bit}_k(i) = 0\},$$

where \bar{X} is the complement of set X .

What remains to be shown is that the operations on the sequence of sets $\langle G_{d-1}, \dots, G_i \rangle$ that reflect changes on the stack of recursive calls of the universal algorithm can indeed be implemented using small numbers of symbolic set operations on the succinct representation $\langle S_1, \dots, S_{\lg d} \rangle$ of the sequence $\langle H_{d-1}, \dots, H_0 \rangle$. We note that there are two types of changes to the sequence $\langle G_{d-1}, \dots, G_i \rangle$ that the universal algorithm makes:

- (a) all components are as before, except for G_i that is replaced by $G_i \setminus B$, for some set $B \subseteq G_i$;
- (b) all components are as before, except that a new entry G_{i-1} is added equal to $G_i \setminus B$, for some set $B \subseteq G_i$.

The corresponding changes to the sequence $\langle H_{d-1}, \dots, H_0 \rangle$ are then:

- (a) all components are as before, except that set H_{i+1} is replaced by $H_{i+1} \cup B$, and set H_i is replaced by $H_i \setminus B$;
- (b) all components are as before, except that set H_i is replaced by B , and set H_{i-1} is replaced by $H_i \setminus B$.

To implement the update of type (a), it suffices to perform the following update to the succinct representation:

$$S'_k = \begin{cases} S_k & \text{if } \text{bit}_k(i+1) = \text{bit}_k(i), \\ S_k \cup B & \text{if } \text{bit}_k(i+1) = 1 \text{ and } \text{bit}_k(i) = 0, \\ S_k \setminus B & \text{if } \text{bit}_k(i+1) = 0 \text{ and } \text{bit}_k(i) = 1. \end{cases}$$

and to implement the update of type (b), it suffices to perform the following:

$$S'_k = \begin{cases} S_k & \text{if } \text{bit}_k(i) = \text{bit}_k(i-1), \\ S_k \setminus (H_i \setminus B) & \text{if } \text{bit}_k(i) = 1 \text{ and } \text{bit}_k(i-1) = 0, \\ S_k \cup (H_i \setminus B) & \text{if } \text{bit}_k(i) = 0 \text{ and } \text{bit}_k(i-1) = 1. \end{cases} \quad \square$$