A Quasi-Polynomial Black-Box Algorithm for Fixed Point Evaluation

André Arnold
Independent Researcher, Talence, France
aa-labri@sfr.fr

Damian Niwiński
Institute of Informatics, University of Warsaw, Poland
niwinski@mimuw.edu.pl

Paweł Parys
Institute of Informatics, University of Warsaw, Poland
parys@mimuw.edu.pl

Abstract

We consider nested fixed-point expressions like $\mu z.\nu y.\mu x.f(x, y, z)$ evaluated over a finite lattice, and ask how many queries to a function f are needed to find the value. The previous upper bounds for a monotone function f of arity d over the lattice $\{0, 1\}^n$ were of the order $n^{\mathcal{O}(d)}$, whereas a lower bound of $\Omega\left(\frac{n^2}{\lg n}\right)$ is known in case when at least one alternation between the least (μ) and the greatest (ν) fixed point occurs in the expression. Following a recent development for parity games, we show here that a quasi-polynomial number of queries is sufficient, namely $n^{\lg(d/\lg n)+\mathcal{O}(1)}$. The algorithm is an abstract version of several algorithms proposed recently by a number of authors, which involve (implicitly or explicitly) the structure of a universal tree. We then show a quasi-polynomial lower bound for the number of queries used by the algorithms in consideration.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Mu-calculus, Parity games, Quasi-polynomial time, Black-box algorithm

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.9

Funding *Paweł Parys*: Author supported by the National Science Centre, Poland (grant no. 2016/22/E/ST6/00041).

1 Introduction

Computing fixed points over a finite lattice is a fundamental problem whose algorithmic nature is not yet completely understood. The problem can be stated as evaluation of an expression

$$\theta_d x_d.\theta_{d-1}x_{d-1}.\cdots.\theta_2 x_2.\theta_1 x_1.f(x_1, x_2, \dots, x_d), \tag{1}$$

where f is a monotonic mapping $f\colon (\{0,1\}^n)^d \to \{0,1\}^n$ for some $n \geq 1$, and where $\theta_1, \dots, \theta_d \in \{\mu, \nu\}$ with μ and ν standing for the least and the greatest fixed point, respectively. If every output bit of the function f is a logical OR or a logical AND of some input bits, the problem is well-known to be equivalent to solving parity games [10, 11] (see, e.g., Arnold and Niwiński [1, Section 4] for an exact statement of this equivalence), and in this form it has attracted much attention since at least 20 years. An abstract formulation was previously considered, e.g., by Browne, Clarke, Jha, Long, and Marrero [4], who made one of the first complexity improvements. These authors also noticed that several algorithms make use only of the structure of fixed points, treating the basic operations as black boxes, and suggested a complexity measure, which in our setting boils down to the following:

▶ **Problem 1.1.** *How many queries to the function f are needed to evaluate Expression (1)?*

The problem has been taken by Parys [24], who showed in particular that at least $\Omega\left(\frac{n^2}{\lg n}\right)$ queries are needed in case when at least one alternation between μ and ν occurs in the expression. One might have expected that this lower bound extends to roughly n^d in general case, but we will see below that it is not the case.

The breakthrough result by Calude, Jain, Khoussainov, Li, and Stephan [5] exhibiting a quasi-polynomial time algorithm for parity games has triggered an intensive flow of research [12, 13, 14, 18, 19, 21, 22, 25], aiming in potential improvement, but also in better understanding of the new complexity situation. The new algorithm has a direct consequence on fixed-point computation: it applies to Expression (1) if the function f is given by a vector of Boolean terms. While these terms can be in general exponential in n and d, Hausmann and Schröder [14] showed that the blow-up can be controlled: Expression (1) can be evaluated in quasi-polynomial time provided that the function f itself can be evaluated in quasi-polynomial time.

In the present paper we focus on the black-box model and evaluate Expression (1) using a quasi-polynomial number of queries to the function f. The algorithm is an abstract version of some recent algorithms for parity games. Our starting point was an algorithm proposed by Parys [25], which is a standard McNaughton–Zielonka's algorithm [23, 29] enhanced by some additional control over recursion. But we also exploit the subsequent improvement by Lehtinen, Schewe, and Wojtczak [22], and a generalisation by Jurdziński and Morvan [19], where the control mechanism uses *universal trees* – a key tool in the modern analysis of parity games [8].

The number of queries to the function f is bounded by $n^{2\cdot\lg(d/\lg n)+\mathcal{O}(1)}$ (in this paper lg stands for the binary logarithm). It can be further improved to $n^{\lg(d/\lg n)+\mathcal{O}(1)}$ (i.e., almost quadratically) by reorganising the algorithm in asymmetric manner, which is a celebrated trick in fixed-point computation invented by Seidl [26]. While the symmetric version of our algorithm resembles the aforementioned recursive algorithms solving parity games [19, 22, 25], the asymmetric version is close to the earlier quasi-polynomial algorithms solving parity games [5, 12, 18, 21]. In consequence, we see a direct link between the two families of algorithms for parity games, which is hardly visible until the μ-calculus perspective is adopted. Finally we show a kind of a lower bound for the class of algorithms in consideration. While this is not an absolute lower bound for the black-box complexity like, e.g., in Parys [24], we show that any algorithm of the considered form must involve a universal tree and therefore, by the result of Czerwiński *et al.* [8], it uses a quasi-polynomial number of queries.

Related work. As mentioned above, we build on recent algorithms for parity games [19, 22, 25]. All these algorithms exploit the concept of *dominion* introduced by Jurdziński, Paterson, and Zwick [20], and our initial step in this paper is a fixed-point interpretation of dominions. The key result on decompositions of dominions is similar to the result on attractor decompositions of dominions considered by Jurdziński and Morvan [19].

The relation of the new techniques for parity games to the μ-calculus has been addressed by Lehtinen [21], who showed, in particular, a new upper bound on the alternation depth of a fixed-point formula. Hausmann and Schröder in the aforementioned work [14] invent a quasi-polynomial time algorithm for computing fixed points of monotone set-valued functions.[1] They did not considered black-box model, but it can be seen that their algorithm (adapted to Expression (1)) performs $n \cdot d^{\lceil \lg n\rceil+2}$ queries to the function f, which is similar to our

[1] After the submission of our paper, Hausmann and Schröder released a new version of their work [15], where they develop a unified method of fixed-point evaluation based on universal graphs [7].

algorithm in its asymmetric version. These authors ask in the conclusion whether this method can also incorporate the algorithm by Parys [25]. For parity games, this question is essentially answered by a meta-algorithm of Jurdziński and Morvan [19], that captures all algorithms known so far including [22, 25]. In our work (that we started not knowing the work of Hausmann and Schröder [14, 15]) we develop a quasi-polynomial method directly for fixed-point evaluation, and additionally show its limitation.

The concept of *symbolic algorithms* considered for parity games by Chatterjee, Dvořák, Henzinger, and Svozil [6], and also by Jurdziński and Morvan [19], is related to Problem 1.1 if the function f is induced by the binary relation of game moves. Then Expression (1) represents the winning region in a parity game [10] (see also [1]), and any black-box algorithm solving Problem 1.1 can be adapted to a symbolic algorithm for parity games.

The complexity of solving parity games is tantalisingly close to polynomial time. As the problem is in $\mathsf{NP} \cap \mathsf{co\text{-}NP}$ (even in $\mathsf{UP} \cap \mathsf{co\text{-}UP}$ [16]), one can hardly expect a lower bound above the P-completeness, which holds already for reachability games [28]. The research in this direction focuses on specific classes of algorithms. The aforementioned (almost) quadratic lower bound by Parys [24] concerns the number of queries used by a black-box algorithm. Recently Czerwiński *et al.* [8] estimated the size of universal trees, which are behind the algorithms exhibiting a *separation scheme* first pinpointed by Bojańczyk and Czerwiński [3]. This gives evidence that the quasi-polynomial complexity of the original algorithm [5] as well as the follow-ups [12, 13, 18, 21] is tight. There has been some hope that the newest approach based on controlling recursion in the McNaughton–Zielonka algorithm [19, 22, 25] may avoid this barrier, but our present results give evidence that it is not the case.

2 Basic concepts

Fixed points. By the celebrated theorem of Knaster and Tarski, if $f\colon L \to L$ is a monotone function over a complete lattice $\langle L, \leq \rangle$, then it has the least (μ) and the greatest (ν) fixed points satisfying the formulae

$$\mu x.f(x) = \inf\{a \mid f(a) \leq a\} \qquad \text{and} \qquad \nu x.f(x) = \sup\{a \mid a \leq f(a)\}\,, \tag{2}$$

respectively. For a monotone function of several arguments, we can apply fixed-point operators successively; for example, $\nu y.\mu x.g(x,y)$ is the greatest fixed point of the mapping $y \mapsto \mu x.g(x,y)$, etc. This gives the semantics of Expression (1).

As it is easy to see that

$$\theta y.\theta x.g(x,y) = \theta x.g(x,x), \tag{3}$$

for $\theta \in \{\mu, \nu\}$, we can without loss of generality assume that the μ and ν operators in Expression (1) alternate.

In this paper, as L we take a finite power $\mathbb{B}^n$ of the Boolean lattice $\mathbb{B} = \{0,1\}$ with the componentwise order denoted by $\leq$. We denote the least and the greatest element of $\mathbb{B}^n$ by $\mathbf{0}$ and $\mathbf{1}$, respectively (assuming that n is clear from the context). We use a semiring notation for join and meet, that is, for $A, B \in \mathbb{B}^n$ we write

$$A + B \overset{\text{def}}{=} \sup(A,B) \qquad \text{and} \qquad A * B \overset{\text{def}}{=} \inf(A,B).$$

Moreover, for $f\colon (\mathbb{B}^n)^d \to \mathbb{B}^n$ and for $A \in \mathbb{B}^n$, let $f^{\upharpoonright A}\colon (\mathbb{B}^n)^{d-1} \to \mathbb{B}^n$ be the mapping defined by

$$f^{\upharpoonright A}(x_1, \dots, x_{d-1}) \overset{\text{def}}{=} f(x_1, \dots, x_{d-1}, A).$$

We refer the reader to Arnold and Niwiński [1] for basic properties of fixed points.

Restrictions. If $A \leq B$, and $f\colon (\mathbb{B}^n)^d \to \mathbb{B}^n$ is monotone in each argument, we let

$$f_{A,B}(x_1, \dots, x_d) \stackrel{\text{def}}{=} A + B * f(x_1, \dots, x_d).$$

Clearly $f_{A,B}$ is monotone as well. Note that $f_{\mathbf{0},\mathbf{1}} = f$.

In this paper we often consider a generalisation of Expression (1) where f is replaced by the restricted function $f_{A,B}$. Namely, for $\Theta = \langle \theta_d, \theta_{d-1}, \dots, \theta_1 \rangle$, with $\theta_i \in \{\mu, \nu\}$, we form an expression $(\Theta, f, (A, B))$ whose value is

$$\|(\Theta, f, (A, B))\| \stackrel{\text{def}}{=} \theta_d x_d.\theta_{d-1} x_{d-1}.\cdots.\theta_1 x_1.f_{A,B}(x_1, \dots, x_d).$$

The object $(\Theta, f, (A, B))$, where the length of Θ equals the number of arguments in the monotone function $f\colon \mathbb{B}^n \to \mathbb{B}^n$ and where $A \leq B$, is called a *fixed-point expression over* $\mathbb{B}^n$ (or simply a *fixed-point expression* if n is clear from the context). We write $|\Theta|_\mu$ and $|\Theta|_\nu$ for the number of μ and ν operators in Θ, respectively.

▶ Remark 2.1. The above restriction allows us to “narrow the scope” of a function f to an interval $[A, B] = \{x \mid A \leq x \leq B\}$ that can be identified with $\mathbb{B}^I$, where $I = \{i \in \{1, \dots, n\} \mid A_i < B_i\}$. It would be perhaps more natural to define it with a function

$$A + B * f(A + x_1 * B, A + x_2 * B, \dots, A + x_n * B);$$

such a function clearly depends only on the bits x_i with $i \in I$. But one can easily prove that $\theta x.A + B * f(A + B * x) = \theta x.A + B * f(x)$, and consequently (by induction) the two definitions lead to the same fixed points. We have chosen $f_{A,B}$ above for its simplicity.

Because $\|(\Theta, f, (A, B))\|$ is a fixed point of $f_{A,B}$, it lies between the bounds A, B:

▶ **Proposition 2.2.** *For every fixed-point expression* $(\Theta, f, (A, B))$,

$$A \leq \|(\Theta, f, (A, B))\| \leq B.$$

Moreover, the value of $(\Theta, f, (A, B))$ does not depend on the bounds A and B, assuming that it lies between these bounds (see Appendix A for a proof):

▶ **Proposition 2.3.** *If* $A \leq C \leq \|(\Theta, f, (A, B))\| \leq D \leq B$*, then*

$$\|(\Theta, f, (A, B))\| = \|(\Theta, f, (C, D))\|.$$

Duality. The *dual* of $b \in \mathbb{B}$ is $\overline{b} = 1 - b$, and the dual of a vector $x = (x_1, \dots, x_n) \in \mathbb{B}^n$ is $\overline{x} = (\overline{x_1}, \dots, \overline{x_n})$. The dual of a function $f\colon (\mathbb{B}^n)^d \to \mathbb{B}^n$ is the function $\widetilde{f}$ defined by $\widetilde{f}(x_1, \dots, x_d) = \overline{f(\overline{x_1}, \dots, \overline{x_d})}$. The dual $\widetilde{\theta}$ of $\theta \in \{\nu, \mu\}$ is the other element of $\{\nu, \mu\}$, and the dual of $\Theta = \langle \theta_1, \dots, \theta_d \rangle$ is $\widetilde{\Theta} \stackrel{\text{def}}{=} \langle \widetilde{\theta_1}, \dots, \widetilde{\theta_d} \rangle$. The dual of $\mathcal{F} = (\Theta, f, (A, B))$ is $\widetilde{\mathcal{F}} \stackrel{\text{def}}{=} (\widetilde{\Theta}, \widetilde{f}, (\overline{B}, \overline{A}))$. The following is a direct consequence of the definition.

▶ **Proposition 2.4.** *For every fixed-point expression* $\mathcal{F}$ *we have* $\|\widetilde{\mathcal{F}}\| = \overline{\|\mathcal{F}\|}$.

This proposition allows us to perform proofs by duality: it is enough to prove statements for one of the fixed-point operators, μ or ν, and then a proof for the other operator follows by considering the dual expression.

Trees. *Ordered trees* (or simply *trees*) are defined by induction: if $T_1, \dots, T_k$ are ordered trees, then $\langle T_1, \dots, T_k \rangle$ is an ordered tree (where possibly $k = 0$, which is the base of the induction). A *node*, a *leaf*, a *child*, a *descendant*, a *parent*, an *ancestor*, etc., are defined as expected; we skip formal definitions. The *width* of a tree T, denoted $|T|$, equals 1 for $T = \langle\rangle$, and $|T_1| + \cdots + |T_k|$ for $T = \langle T_1, \dots, T_k \rangle$ with $k \geq 1$ (we can identify the width with the number of leaves of a tree). The *height* of a tree T equals 0 for $T = \langle\rangle$, and 1 plus the maximum of heights of $T_1, \ldots, T_k$ for $T = \langle T_1, \ldots, T_k\rangle$ with $k \geq 1$. We allow concatenation of trees, so that $\langle T_1, \ldots, T_k\rangle \cdot \langle T_{k+1}, \ldots, T_p\rangle$ amounts to $\langle T_1, \ldots, T_k, T_{k+1}, \ldots, T_p\rangle$, and T^n abbreviates $\underbrace{T \cdot \ldots \cdot T}_{n}$.

A tree is *equitable* if all its branches have the same length; more formally, $T = \langle T_1, \ldots, T_k\rangle$ is equitable if all $T_1, \ldots, T_k$ are equitable and have the same height. In the sequel, we almost exclusively consider equitable trees. The *level* of a node in an equitable tree of height h is its distance from the leaves (in particular leaves are at level 0, and the root is at level h).

Intuitively, a tree T *embeds* in a tree U if T can be obtained from U by pruning some subtrees. More formally, $T = \langle T_1, \ldots, T_k\rangle$ embeds in $U = \langle U_1, \ldots, U_p\rangle$ if there exist indices $j_1, \ldots, j_k$ such that $1 \leq j_1 < \cdots < j_k \leq p$ and T_i embeds in U_{j_i} for all $i \in \{1, \ldots, k\}$. (Thus $\langle\rangle$ embeds in every tree.)

A tree U is (n, h)*-universal* if it is equitable, has height h, and every (equitable) tree T of height at most h and width at most n embeds in U. In this statement it does not matter whether or not we require that T is equitable, because every tree embeds in some equitable tree of the same height and width. We know three families of (n, h)-universal trees:

$$\begin{aligned}
&C_{n,0} = P_{n,0} = S_{n,0} = S_{0,h} = \langle\rangle, && \\
&C_{n,h} = \langle C_{n,h-1}\rangle^n && \text{for } h \geq 1, \\
&P_{n,h} = \langle P_{\lfloor n/2\rfloor,h-1}\rangle^{\lfloor n/2\rfloor} \cdot \langle P_{n,h-1}\rangle \cdot \langle P_{\lfloor n/2\rfloor,h-1}\rangle^{\lfloor n/2\rfloor} && \text{for } h \geq 1, \\
&S_{n,h} = S_{\lfloor n/2\rfloor,h} \cdot \langle S_{n,h-1}\rangle \cdot S_{\lfloor n/2\rfloor,h} && \text{for } n, h \geq 1.
\end{aligned}$$

▶ **Proposition 2.5** ([19, Proposition 3.2])**.** *Trees $C_{n,h}$, $P_{n,h}$, and $S_{n,h}$ are (n, h)-universal.*

Trees $P_{n,h}$ and $S_{n,h}$ are of quasi-polynomial width; more precisely, they have, respectively, $n^{\lg n+\lg(h/\lg n)+O(1)}$ and $n^{\lg(h/\lg n)+O(1)}$ leaves [18, 25].

3 Algorithm

3.1 Symmetric version

In this section we present an algorithm that computes the value of a fixed-point expression $\mathcal{F}$. To this end, we define a value $\|\mathcal{F}\|_{U,V}$ parameterised by two trees U, V. This value can be computed using $|U| \cdot |V|$ queries to f (cf. Lemma 3.2). Simultaneously, if U and V are universal, this value actually equals $\|\mathcal{F}\|$ (cf. Lemma 3.1). Later, we also prove that this is a necessary condition: the above equality holds only when the trees U and V are universal (cf. Theorem 5.1).

Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression, and let U and V be equitable trees of height, respectively, $|\Theta|_\mu$ and $|\Theta|_\nu$. We define a value $\|\mathcal{F}\|_{U,V}$ by induction on the length of Θ:

(1) if $\Theta = \langle\rangle$, as $\|\mathcal{F}\|_{U,V}$ we take $\|\mathcal{F}\|$, that is, $A + B * f()$;

(2) if $\Theta = \langle\nu\rangle \cdot \Theta'$ and $V = \langle V_1, \ldots, V_p\rangle$, then we take $B_0 = B$, and

$$B_j = \|(\Theta', f^{\upharpoonright B_{j-1}}, (A, B_{j-1}))\|_{U,V_j}$$

for $j \in \{1, \ldots, p\}$, and $\|\mathcal{F}\|_{U,V} = B_p$;

(3) if $\Theta = \langle\mu\rangle \cdot \Theta'$ and $U = \langle U_1, \ldots, U_p\rangle$, then we take $A_0 = A$, and

$$A_j = \|(\Theta', f^{\upharpoonright A_{j-1}}, (A_{j-1}, B))\|_{U_j,V}$$

for $j \in \{1, \ldots, p\}$, and $\|\mathcal{F}\|_{U,V} = A_p$.

▶ **Lemma 3.1.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed point expression, and let U and V be $(n, |\Theta|_\mu)$- and $(n, |\Theta|_\nu)$-universal trees, respectively. Then $\|\mathcal{F}\|_{U,V} = \|\mathcal{F}\|$.*

The above lemma is proven in Section 4. We can easily compute $\|\mathcal{F}\|_{U,V}$, following directly its definition:

▶ **Lemma 3.2.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression, and let U and V be equitable trees of height, respectively, $|\Theta|_\mu$ and $|\Theta|_\nu$. There is an algorithm that computes $\|\mathcal{F}\|_{U,V}$ using $|U| \cdot |V|$ queries to the function f.*

Proof. The proof goes by induction on the length of Θ. A single query is asked in Case (1) of the definition of $\|\mathcal{F}\|_{U,V}$, when all parameters of the original function f have been instantiated, so the claim is true. In Case (2), when $V = \langle V_1, \dots, V_p\rangle$ (where $p \geq 1$, because V has height $|\Theta|_\mu \geq 1$), the number of queries needed to compute $\|\mathcal{F}\|_{U,V}$ amounts to the sum of the analogous numbers for $\|(\Theta', f^{\restriction B_{j-1}}, (A, B_{j-1}))\|_{U,V_j}$, which, by the induction hypothesis, equals

$$|U| \cdot |V_1| + \cdots + |U| \cdot |V_p| = |U| \cdot |V|.$$

Case (3) is similar. ◀

▶ Remark 3.3. The above direct "naive" algorithm computing $\|\mathcal{F}\|_{U,V}$ can be made "adaptive": whenever in Case (2) we have that $B_{j-1} = B_{j-2}$ and $V_j = V_{j-1}$, we do not need to compute B_j as $\|(\Theta', f^{\restriction B_{j-1}}, (A, B_{j-1}))\|_{U,V_j}$, but we can simply take $B_j = B_{j-1}$, saving some number of queries to the function f; likewise in Case (3).

Lemma 3.1 combined with Lemma 3.2 and with the estimation of the width of the universal tree $S_{n,h}$ recalled after Proposition 2.5, yields a possible answer to Problem 1.1: $n^{2\cdot\lg(d/\lg n)+O(1)}$ queries to the function f are enough to evaluate Expression (1). In the next section we improve this complexity (almost) quadratically.

The above algorithm can be seen as a translation of the generic recursive algorithm of Jurdziński and Morvan [19] solving parity games (which is also parameterised by two trees assumed to be universal) to the setting of μ-calculus. There is one difference: the algorithm for parity games includes computation of attractors, which is absent here. The above algorithm used with trees $S_{n,h}$ resembles the recursive algorithm of Lehtinen, Schewe, and Wojtczak [22], and the adaptive version of the algorithm (cf. Remark 3.3) used with trees $P_{n,h}$ resembles the recursive algorithm of Parys [25]. The adaptive version of the algorithm used with the complete trees $C_{n,h}$ gives a version of the naive-iteration algorithm that works in polynomial time assuming that n is fixed (cf. Parys [24]).

3.2 Asymmetric version

We now modify the algorithm from the previous section using an idea of Seidl [26]. As a first step, we define yet another value, $\|\mathcal{F}\|_V$, parameterised by a single tree V. In its definition, we proceed in an asymmetric way: on the μ side we simply compute all fixed points, and on the ν side we follow a structure of the tree V. It turns out that $\|\mathcal{F}\|_V$ equals $\|\mathcal{F}\|$ if V is universal (cf. Lemma 3.4 below). There is however no direct analogue to Lemma 3.2, and keeping the number of queries low requires some care. We explore the fact that nested applications of fixed points of the same kind (μ in this case) can be reduced to a single application over a vector of variables in a system of equations, which is precisely the idea behind Seidl's algorithm [26]. Thus, as a second step of our algorithm, we replace our recursive definition of $\|\mathcal{F}\|_V$ by an equivalent system of least fixed-point equations. This system has size proportional to $|V|$, and thus can be solved using such a number of queries to the function f (see Lemma 3.6 below), yielding the value $\|\mathcal{F}\|$.

Algorithm 1

```
procedure $\textsc{Generate}(x, \Theta, \mathbf{f}, \mathbf{B}, V)$
begin
    if $\Theta = \langle\rangle$ then
        output "$x = \mathbf{B} * \mathbf{f}$";
    if $\Theta = \langle\nu\rangle \cdot \Theta'$ and $V = \langle V_1, \dots, V_p\rangle$ then begin
        $\mathbf{B}_0 = \mathbf{B}$;
        $\mathbf{B}_p = x$;
        for $j = 1$ to $p - 1$ do
            $\mathbf{B}_j = \textsc{FreshVariable}()$;
        for $j = 1$ to $p$ do
            $\textsc{Generate}(\mathbf{B}_j, \Theta', \mathbf{f}^{\restriction \mathbf{B}_{j-1}}, \mathbf{B}_{j-1}, V_j)$;
    end;
    if $\Theta = \langle\mu\rangle \cdot \Theta'$ then
        $\textsc{Generate}(x, \Theta', \mathbf{f}^{\restriction x}, \mathbf{B}, V)$;
end;
$x_{res} = \textsc{FreshVariable}()$;
$\textsc{Generate}(x_{res}, \Theta, \mathsf{f}(?, \dots, ?), \mathsf{x}_0, V)$;
```

Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression, and let V be an equitable tree of height $|\Theta|_\nu$. We define $\|\mathcal{F}\|_V$ by induction on the length of Θ:

(1) if $\Theta = \langle\rangle$, as $\|\mathcal{F}\|_V$ we take $\|\mathcal{F}\|$, that is, $A + B * f()$;

(2) if $\Theta = \langle\nu\rangle \cdot \Theta'$ and $V = \langle V_1, \dots, V_p\rangle$, then we take $B_0 = B$, and

$$B_j = \|(\Theta', f^{\restriction B_{j-1}}, (A, B_{j-1}))\|_{V_j}$$

for $j \in \{1, \dots, p\}$, and $\|\mathcal{F}\|_V = B_p$;

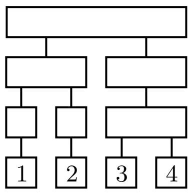
(3) if $\Theta = \langle\mu\rangle \cdot \Theta'$, then we take $\|\mathcal{F}\|_V = \mu x.\|(\Theta', f^{\restriction x}, (A, B))\|_V$ (i.e., the least fixed point of the mapping $x \mapsto \|(\Theta', f^{\restriction x}, (A, B))\|_V$).

When we iterate a function n times, starting from the least element (as in the definition of $\|\mathcal{F}\|_{C_{n,h},V}$), we reach the least fixed point (appearing in the definition of $\|\mathcal{F}\|_V$). It is thus not difficult to prove the following lemma, saying that instead of computing $\|\mathcal{F}\|$ we can compute $\|\mathcal{F}\|_V$ for some universal tree V (see Appendix B for more details).

▶ **Lemma 3.4.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression, and let V be an equitable tree of height $|\Theta|_\nu$. Then $\|\mathcal{F}\|_V = \|\mathcal{F}\|_{C_{n,|\Theta|_\mu},V}$. In particular, if V is $(n, |\Theta|_\nu)$-universal, then (by Proposition 2.5 and Lemma 3.1) $\|\mathcal{F}\|_V = \|\mathcal{F}\|$.*

Next, we consider a system of equations corresponding to the definition of $\|\mathcal{F}\|_V$. For simplicity, we assume here that $A = \mathbf{0}$ and $B = \mathbf{1}$, that is, we consider $\mathcal{F} = (\Theta, f, (\mathbf{0}, \mathbf{1}))$. Let $\mathcal{V}$ be a set of variables, and let $\mathcal{V}_\mathbf{1} = \mathcal{V} \uplus \{\mathsf{x}_0\}$ contain additionally a variable (constant) x_0 that is always valuated to the element $\mathbf{1}$ of $\mathbb{B}^n$. Our equations will be of the form $x = y_0 * \mathsf{f}(y_1, \dots, y_d)$, where $x \in \mathcal{V}$, $y_0, \dots, y_d \in \mathcal{V}_\mathbf{1}$, and f is a constant denoting the considered function.

The system of equations is generated by Algorithm 1. In this algorithm, $\mathbf{f}$ is an expression of the form $\mathsf{f}(?, \dots, ?, y_{k+1}, \dots, y_d)$ for some variables $y_{k+1}, \dots, y_d \in \mathcal{V}_\mathbf{1}$. Moreover, $\mathbf{f}^{\restriction z}$ for $z \in \mathcal{V}_\mathbf{1}$ denotes $\mathsf{f}(?, \dots, ?, z, y_{k+1}, \dots, y_d)$ (we substitute z for the last question mark). The parameter $\mathbf{B}$ of the procedure $\textsc{Generate}$ is an element of $\mathcal{V}_\mathbf{1}$. Note that the algorithm is



$$\begin{aligned}
\mathsf{x}_1 &= \mathsf{x}_0 * \mathsf{f}(\mathsf{x}_1,\mathsf{x}_0,\mathsf{x}_1,\mathsf{x}_0,\mathsf{x}_2,\mathsf{x}_0,\mathsf{x}_4),\\
\mathsf{x}_2 &= \mathsf{x}_1 * \mathsf{f}(\mathsf{x}_2,\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_0,\mathsf{x}_4),\\
\mathsf{x}_3 &= \mathsf{x}_2 * \mathsf{f}(\mathsf{x}_3,\mathsf{x}_2,\mathsf{x}_4,\mathsf{x}_2,\mathsf{x}_4,\mathsf{x}_2,\mathsf{x}_4),\\
\mathsf{x}_4 &= \mathsf{x}_3 * \mathsf{f}(\mathsf{x}_4,\mathsf{x}_3,\mathsf{x}_4,\mathsf{x}_2,\mathsf{x}_4,\mathsf{x}_2,\mathsf{x}_4).
\end{aligned}$$

Figure 1 A tree V (left), and the system of equations generated for $\|(\langle\mu\rangle\cdot\langle\nu,\mu\rangle^3, f, (\mathbf{0},\mathbf{1}))\|_V$ (right). Variable x_4 stores the result. Observe a correspondence between variables and leaves of V.

syntactical: it depends on the tree V and on the sequence Θ, but not on any particular interpretation of f. Notice that when the algorithm enters line 4 (i.e., it is going to output an equation), $\mathbf{f}$ contains no ?'s. See also Appendix C for another definition of the system.

For an example of a generated system of equations, see Figure 1.

Now, given $\mathcal{F} = (\Theta, f, (A, B))$ and V like in the definition of $\|\mathcal{F}\|_V$, where additionally $A = \mathbf{0}$ and $B = \mathbf{1}$, we can interpret and solve the resulting system in the lattice $\mathbb{B}^n$ with f interpreted by the function f. Thanks to a direct correspondence between the system and the recursive definition of $\|\mathcal{F}\|_V$, we obtain the following lemma (see Appendix D for a tedious but straightforward proof).

▶ **Lemma 3.5.** *Let $\mathcal{F} = (\Theta, f, (\mathbf{0},\mathbf{1}))$ be a fixed-point expression, and let V be an equitable tree of height $|\Theta|_\nu$. The value assigned to the variable x_{res} in the least solution of the system of equations generated by Algorithm 1 equals $\|\mathcal{F}\|_V$.*

We are now ready to state an analogue to Lemma 3.2.

▶ **Lemma 3.6.** *Let $\mathcal{F} = (\Theta, f, (\mathbf{0},\mathbf{1}))$ be a fixed-point expression, and let V be an equitable tree of height $|\Theta|_\nu$. There is an algorithm that computes $\|\mathcal{F}\|_V$ using between $|V|$ and $|V|\cdot(1+nd)$ queries to the function f.*

Proof. By Lemma 3.5, it is enough to generate a system of equations using Algorithm 1, and then to find the least solution of this system, having $|V|$ equations. The least solution can be found by a standard worklist algorithm (cf. [26, Proposition 2]). In this algorithm, we keep updating an underapproximation of the least solution, stage by stage, until reaching a fixed point. In every stage, we re-evaluate right sides only of those equations in which at least one variable was modified in the previous stage, and we store results in variables appearing on the left side (before the first stage we set x_0 to $\mathbf{1}$, other variables to $\mathbf{0}$, and we treat all of them as modified). Each among d arguments of $|V|$ equations can be updated (increased) at most n times, yielding at most $|V|\cdot(1+nd)$ (and at least $|V|$) queries to the function f. ◀

For $V = S_{n,|\Theta|_\nu}$, the number of queries becomes $n^{\lg(d/\lg n)+O(1)}$, so (almost) quadratically better than for the algorithm of Section 3.1, and, to our knowledge, the best so far.

▶ **Corollary 3.7.** *There is an algorithm to evaluate the Expression (1) using at most $n^{\lg(d/\lg n)+O(1)}$ queries.*

The reduction from an expression using nested μ and ν fixed-point operators to a system of equations involving only the least fixed point can be compared to a reduction from parity games to reachability (or safety) games. As explained in Czerwiński *et al.* [8], such a reduction stands behind the "iterative" quasi-polynomial algorithms [5, 12, 18, 21], but (for universal trees of exponential width) is present also in earlier results [2, 17].

3.3 Time and space complexity

Although our main interest is in the number of queries to the function f performed by our algorithms, let us also analyse their time and space complexity. Let t_f be the time needed to answer a single query (i.e., to compute the value of f for given arguments), and, as previously, let n and d denote, respectively, the height of the considered lattice and the arity of f.

The symmetric version of our algorithm spends time $t_f \cdot n^{2\cdot\lg(d/\lg n)+O(1)}$ on computing values of the function f, and beside of that performs some recursive calls following the structure of universal trees $S_{n,h}$ (of course there is no need to actually construct these trees). Formally, the number of recursive calls depends on the number of nodes of the two trees, not on the number of leaves. However, in every tree, the number of nodes with at least two children is smaller than the number of leaves, and one can easily improve the algorithm so that it "skips" nodes with a single child. Thus, the time spent on performing recursive calls can be ignored. The memory usage is $O(n \cdot d)$ (the depth of the recursion is d, and on every level it is enough to store a constant number of elements of $\mathbb{B}^n$).

In the asymmetric version, the system of equations can be solved (using the method described in the proof of Lemma 3.6) in time proportional to the number of queries. Moreover, we do not need to actually generate the system; we can instead describe it explicitly based on the considered tree (see Appendix C for details). Such a description allows to navigate in the system (e.g. to find all equations containing a given variable) with a constant overhead. Thus, due to Corollary 3.7, the running time is $t_f \cdot n^{\lg(d/\lg n)+O(1)}$. To compare, the algorithm of Hausmann and Schröder [14] needs a factor $O(n \cdot d^{\lceil \lg n \rceil + 2})$ per query, not $O(1)$.

Concerning the memory usage, in a straightforward implementation we store a value for every variable while solving the system of equations. We can do better, however. Indeed, we can observe that if we write the i-th equation as

$$\mathsf{x}_i = \mathsf{x}_{k(i,0)} * \mathsf{f}(\mathsf{x}_{k(i,1)}, \ldots, \mathsf{x}_{k(i,d)}),$$

then for all $j \in \{0, \ldots, d\}$ we have $k(i,j) \le k(i',j)$ whenever $i \le i'$ (this is best visible while looking at the explicit description of the system, introduced in Appendix C). Thanks to this monotonicity of the system, and monotonicity of f, values assigned to the variables after every stage of the algorithm satisfy $\mathsf{x}_i \ge \mathsf{x}_{i'}$ whenever $i \le i'$ (this is satisfied before the first stage, and is then preserved). Such a valuation can be stored very succinctly, using n numbers: for every bit of $\mathbb{B}^n$ it is enough to remember the number of the last variable in which this bit is set to 1. Going further: in order to remember which variables were modified in the last stage, it is enough to remember two last valuations of the variables. It is not difficult to adapt the algorithm to such a representation of valuations. Thus, the memory usage can be reduced to $O(n)$, modulo a polylogarithmic factor.

4 Correctness of the algorithms

In this section we prove that if U and V are universal trees, then $\|\mathcal{F}\| = \|\mathcal{F}\|_{U,V}$ (Lemma 3.1). As a first step, we introduce *sup-dominions* and their *decompositions*, and we prove some properties of these notions. Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression. A value $D \in \mathbb{B}^n$ such that $A \le D \le B$ is a *sup-dominion* for $\mathcal{F}$ if $D = \|(\Theta, f, (A, D))\|$.

For readers familiar with game-theoretic dominions considered in prior work [19, 20, 22, 25], the following analogy may be useful: a sup-dominion for $(\Theta, f, (A, B))$ is an area D where player Even can force the play either to reach A or to ensure the parity condition while not leaving D. One can also define inf-dominions (by requiring $D = \|(\Theta, f, (D, B))\|$), describing the complement of an analogous area for player Odd; however, we conduct our correctness proof using only sup-dominions, and then arguing by duality.

We first note some useful properties analogous to the properties of game dominions noted in prior work [19, 22, 25], stemming from Proposition 2.3.

▶ **Lemma 4.1.** *For every fixed-point expression $\mathcal{F}$, the value $\|\mathcal{F}\|$ is a sup-dominion for $\mathcal{F}$.*

▶ **Lemma 4.2.** *If D is a sup-dominion for a fixed-point expression $(\Theta, f, (A, B))$, and $A \leq A' \leq D$, then D is also a sup-dominion for $(\Theta, f, (A', B))$.*

Lemmata 4.1 and 4.2 are immediate consequences of definitions and of Proposition 2.3.

▶ **Lemma 4.3.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression with $\Theta = \langle\theta\rangle \cdot \Theta'$. If D is a sup-dominion for $\mathcal{F}$, then D is also a sup-dominion for $(\Theta', f^{\upharpoonright D}, (A, D))$.*

Proof. We have by definition $D = \|(\Theta, f, (A, D))\| = \theta x.\|(\Theta', f^{\upharpoonright x}, (A, D))\|$. This means that D is a fixed point of the mapping $x \mapsto \|(\Theta', f^{\upharpoonright x}, (A, D))\|$, that is, $D = \|(\Theta', f^{\upharpoonright D}, (A, D))\|$, as required. ◀

▶ **Lemma 4.4.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression with $\Theta = \langle\mu\rangle \cdot \Theta'$. For every sup-dominion D for $\mathcal{F}$ such that $A < D$ there exists a sup-dominion D' for $(\Theta', f^{\upharpoonright A}, (A, B))$ such that $A < D' \leq D$.*

Proof. Let $g(x) = \|(\Theta', f^{\upharpoonright x}, (A, D))\|$ so that $D = \|(\Theta, f, (A, D))\| = \mu x.g(x)$. We have, by monotonicity, $D = g(D) \geq g(A) = \|(\Theta', f^{\upharpoonright A}, (A, D))\|$. Hence, from Proposition 2.3 we obtain $g(A) = \|(\Theta', f^{\upharpoonright A}, (A, g(A)))\|$. That is, $g(A)$ is a sup-dominion for $(\Theta', f^{\upharpoonright A}, (A, B))$, which moreover satisfies $A \leq g(A) \leq D$ (where $A \leq g(A)$ is by Proposition 2.2). If $g(A) = A$, we would have $D = \mu x.g(x) \leq A$, a contradiction. Hence $g(A) > A$, and $D' = g(A)$ satisfies the claim. ◀

Next, we define the crucial concept of the paper: dominion decompositions. Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression. The definition is by induction on the length of Θ. A pair $(D, \mathcal{H})$ is a *sup-dominion decomposition* for $\mathcal{F}$ if D is a sup-dominion for $\mathcal{F}$ and

- if $\Theta = \langle\rangle$, then $\mathcal{H} = \langle\rangle$;
- if $\Theta = \langle\nu\rangle \cdot \Theta'$, then $(D, \mathcal{H})$ is a sup-dominion decomposition for $(\Theta', f^{\upharpoonright D}, (A, D))$;
- if $\Theta = \langle\mu\rangle \cdot \Theta'$, then $\mathcal{H} = \langle(D_1, \mathcal{H}_1), \dots, (D_k, \mathcal{H}_k)\rangle$, where, assuming $D_0 = A$, for every $i \in \{1, \dots, k\}$ the pair $(D_i, \mathcal{H}_i)$ is a sup-dominion decomposition for $(\Theta', f^{\upharpoonright D_{i-1}}, (D_{i-1}, B))$, and $D_k = D$.

To a sup-dominion decomposition $(D, \mathcal{H})$ we can assign a tree $T_{\mathcal{H}}$ by forgetting the dominions stored in the decomposition and taking only its "shape": for $\mathcal{H} = \langle(D_1, \mathcal{H}_1), \dots, (D_k, \mathcal{H}_k)\rangle$ we let $T_{\mathcal{H}} = \langle T_{\mathcal{H}_1}, \dots, T_{\mathcal{H}_k}\rangle$.

The following crucial lemma states that every dominion has a decomposition (not necessarily unique). Here by $|D - A|_1$ (for $D \geq A$) we denote the number of bits that are 1 in D but 0 in A.

▶ **Lemma 4.5.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression. For every sup-dominion D for $\mathcal{F}$ such that $A < D$ there exists a sup-dominion decomposition $(D, \mathcal{H})$ for $\mathcal{F}$ such that $T_{\mathcal{H}}$ is an equitable tree of height $|\Theta|_\mu$ and of width at most $|D - A|_1$.*

Proof. The proof is by induction on the length of Θ. For $\Theta = \langle\rangle$ we just take $\mathcal{H} = \langle\rangle$ (notice that $T_{\mathcal{H}}$ has width $1 \leq |D - A|_1$, because $A < D$). For $\Theta = \langle\nu\rangle \cdot \Theta'$, by Lemma 4.3 D is also a sup-dominion for $(\Theta', f^{\upharpoonright D}, (A, D))$, and thus the sup-dominion decomposition exists by the induction hypothesis.

Finally, suppose that $\Theta = \langle\mu\rangle \cdot \Theta'$. We first construct a sequence $A = D_0 < D_1 < \cdots < D_k \leq D$ such that D_i is a sup-dominion for $(\Theta', f^{\upharpoonright D_{i-1}}, (D_{i-1}, B))$, for every $i \in \{1, \dots, k\}$. We start with $k = 0$ (and $D_0 = A$). Then, as long as $D_k < D$, we create D_{k+1} as follows. First, because D is a sup-dominion for $\mathcal{F}$ and because $A \leq D_k$, by Lemma 4.2 we have that D is a sup-dominion for $(\Theta, f, (D_k, B))$. Second, by Lemma 4.4 (applied to D and $(\Theta, f, (D_k, B))$) there exists sup-dominion D_{k+1} for $(\Theta', f^{\upharpoonright D_k}, (D_k, B))$ such that $D_k < D_{k+1} \leq D$. We can thus attach this D_{k+1} to the sequence. We end the construction when $D_k = D$. To finish the proof, we use the induction hypothesis, which says that every D_i can be extended to a sup-dominion decomposition $(D_i, \mathcal{H}_i)$ for $(\Theta', f^{\upharpoonright D_{i-1}}, (D_{i-1}, B))$ such that $T_{\mathcal{H}_i}$ is an equitable tree of height $|\Theta|_\mu - 1$ and of width at most $|D_i - D_{i-1}|_1$. Then $\mathcal{H} = \langle (D_1, \mathcal{H}_1), \dots, (D_k, \mathcal{H}_k)\rangle$ satisfies the thesis. ◀

Coming slowly to the proof of Lemma 3.1, let us state some basic properties of the value $\|\mathcal{F}\|_{U,V}$. The first two lemmata can be shown by a straightforward induction on the length of Θ:

▶ **Lemma 4.6.** *For every fixed-point expression $(\Theta, f, (A, B))$ and for all equitable trees U, V of height, respectively, $|\Theta|_\mu$ and $|\Theta|_\nu$,*

$$A \leq \|(\Theta, f, (A, B))\|_{U,V} \leq B.$$

▶ **Lemma 4.7.** *For every fixed-point expression $\mathcal{F} = (\Theta, f, (A, B))$ and for all equitable trees U, V of height, respectively, $|\Theta|_\mu$ and $|\Theta|_\nu$, we have $\|\widetilde{\mathcal{F}}\|_{V,U} = \overline{\|\mathcal{F}\|_{U,V}}$.*

Moreover, again by a straightforward induction, it follows that $\|(\Theta, f, (A, B))\|_{U,V}$ is monotone in f, A, and B. This value is also monotone in U and V, in the following sense:

▶ **Lemma 4.8.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression, let T, U be equitable trees of height $|\Theta|_\mu$, and let T', V be equitable trees of height $|\Theta|_\nu$. If T embeds in U, and T' embeds in V, then $\|\mathcal{F}\|_{T,V} \leq \|\mathcal{F}\|_{U,V} \leq \|\mathcal{F}\|_{U,T'}$.*

Proof. It is enough to prove the inequality $\|\mathcal{F}\|_{T,V} \leq \|\mathcal{F}\|_{U,V}$; the second inequality follows then by duality (using Lemma 4.7). The proof is by induction on the length of Θ. For empty Θ both values, $\|\mathcal{F}\|_{T,V}$ and $\|\mathcal{F}\|_{U,V}$ are defined as $A + B * f()$; we have equality. For $\Theta = \langle\nu\rangle \cdot \Theta'$, the inductive definitions of $\|\mathcal{F}\|_{T,V}$ and $\|\mathcal{F}\|_{U,V}$ are the same (we descend recursively using the tree V), so it is enough to use the induction hypothesis and monotonicity. Suppose that $\Theta = \langle\mu\rangle \cdot \Theta'$, and $T = \langle T_1, \dots, T_k\rangle$, and $U = \langle U_1, \dots, U_p\rangle$. Then $\|\mathcal{F}\|_{T,V}$ and $\|\mathcal{F}\|_{U,V}$ are defined as A'_k and A_p, respectively, where

$$\begin{aligned}
&A'_0 = A_0 = A, \\
&A'_i = \|(\Theta', f^{\upharpoonright A'_{i-1}}, (A'_{i-1}, B))\|_{T_i,V} && \text{for } i \in \{1, \dots, k\}, \text{ and} \\
&A_j = \|(\Theta', f^{\upharpoonright A_{j-1}}, (A_{j-1}, B))\|_{U_j,V} && \text{for } j \in \{1, \dots, p\}.
\end{aligned}$$

Observe that if T_i embeds in U_j, and $A'_{i-1} \leq A_{j-1}$, then by monotonicity and by the induction hypothesis we obtain that $A'_i \leq A_j$. Moreover, always $A_{j-1} \leq A_j$, by Lemma 4.6 (this way, we can skip subtrees U_j to which no T_i needs to be embedded). Using these inequalities and the definition of embedding we easily obtain the required thesis $A'_k \leq A_p$. ◀

The next lemma is crucial in the proof of Lemma 3.1.

▶ **Lemma 4.9.** *Let $\mathcal{F} = (\Theta, f, (A, B))$ be a fixed-point expression, and let V be an equitable tree of height $|\Theta|_\nu$. If $(D, \mathcal{H})$ is a sup-dominion decomposition for $\mathcal{F}$, then $D \leq \|\mathcal{F}\|_{T_\mathcal{H},V}$.*

Proof. The proof is by induction on the length of Θ. If $\Theta = \langle\rangle$, we have (by the definition of a sup-dominion) $D = \|(\Theta, f, (A, D))\| = A + D * f() \leq A + B * f() = \|\mathcal{F}\|_{T_\mathcal{H},V}$.

Suppose that $\Theta = \langle\nu\rangle \cdot \Theta'$ and $V = \langle V_1, \dots, V_p\rangle$. Recall the values B_j from the definition of $\|\mathcal{F}\|_{T_\mathcal{H},V}$ (page 5). We prove by an internal induction on $j \in \{0, \dots, p\}$ that $D \leq B_j$ (this gives the thesis since $\|\mathcal{F}\|_{T_\mathcal{H},V} = B_p$). For $j = 0$ the thesis is clear: $D \leq B = B_0$. Next, for $j > 0$ we prove $D \leq B_j$ assuming $D \leq B_{j-1}$. Recall that $(D, \mathcal{H})$ is a sup-dominion decomposition for $(\Theta', f^{\upharpoonright D}, (A, D))$. Thus, the induction hypothesis implies that $D \leq \|(\Theta', f^{\upharpoonright D}, (A, D))\|_{T_\mathcal{H},V_j}$. Using the assumption $D \leq B_{j-1}$ and monotonicity, we obtain that $D \leq \|(\Theta', f^{\upharpoonright B_{j-1}}, (A, B_{j-1}))\|_{T_\mathcal{H},V_j} = B_j$, as required.

Finally, suppose that $\Theta = \langle\mu\rangle \cdot \Theta'$ and $\mathcal{H} = \langle(D_1, \mathcal{H}_1), \dots, (D_k, \mathcal{H}_k)\rangle$. Recall the values A_j from the definition of $\|\mathcal{F}\|_{T_\mathcal{H},V}$. We prove by an internal induction on $j \in \{0, \dots, k\}$ that $D_j \leq A_j$ (this gives the thesis since $\|\mathcal{F}\|_{T_\mathcal{H},V} = A_k$). For $j = 0$ the thesis is clear: $D_0 = A = A_0$. Next, for $j > 0$ we prove $D_j \leq A_j$ assuming $D_{j-1} \leq A_{j-1}$. Recall that $(D_j, \mathcal{H}_j)$ is a sup-dominion decomposition for $(\Theta', f^{\upharpoonright D_{j-1}}, (D_{j-1}, B))$. Thus, the induction hypothesis implies that $D_j \leq \|(\Theta', f^{\upharpoonright D_{j-1}}, (D_{j-1}, B))\|_{T_{\mathcal{H}_j},V}$. Using the assumption $D_{j-1} \leq A_{j-1}$ and monotonicity, we obtain that $D_j \leq \|(\Theta', f^{\upharpoonright A_{j-1}}, (A_{j-1}, B))\|_{T_{\mathcal{H}_j},V} = A_j$, as required. ◀

Proof of Lemma 3.1. Recall that we prove $\|\mathcal{F}\|_{U,V} = \|\mathcal{F}\|$, where $\mathcal{F} = (\Theta, f, (A, B))$ and U, V are universal. It is actually enough to prove $\|\mathcal{F}\| \leq \|\mathcal{F}\|_{U,V}$, because then the converse inequality follows by duality (using Proposition 2.4 and Lemma 4.7). By Lemma 4.1 $\|\mathcal{F}\|$ is a sup-dominion for $\mathcal{F}$. If $\|\mathcal{F}\| = A$, then clearly $\|\mathcal{F}\| \leq \|\mathcal{F}\|_{U,V}$, by Lemma 4.6. Otherwise $\|\mathcal{F}\| > A$, so by Lemma 4.5 there exists a sup-dominion decomposition $(\|\mathcal{F}\|, \mathcal{H})$ for $\mathcal{F}$ such that $T_\mathcal{H}$ is an equitable tree of height $|\Theta|_\mu$ and width at most $|\|\mathcal{F}\| - A|_1 \leq n$. It follows that $T_\mathcal{H}$ embeds in U, and thus $\|\mathcal{F}\| \leq \|\mathcal{F}\|_{T_\mathcal{H},V} \leq \|\mathcal{F}\|_{U,V}$ by Lemmata 4.8 and 4.9. ◀

5 Lower bound

We now present a lower bound for the number of queries used in our method. To this end, we prove that our algorithms work only if they are driven by universal trees:

▶ **Theorem 5.1.** *Let U, V be equitable trees of height h.*

(1) *If $\|\mathcal{F}\|_{U,V} = \|\mathcal{F}\|$ for all fixed-point expressions over $\mathbb{B}^n$ of the form $\mathcal{F} = (\Theta, f, (\mathbf{0}, \mathbf{1}))$, where $|\Theta|_\mu = |\Theta|_\nu = h$, then U and V are (n, h)-universal.*

(2) *If $\|\mathcal{F}\|_V = \|\mathcal{F}\|$ for all fixed-point expressions as above, then V is (n, h)-universal.*

To estimate the number of queries used by the algorithms of Section 3, we combine Theorem 5.1 with a result by Czerwiński *et al.* [8, Theorem 2.3] saying that any (n, h)-universal tree has at least $\binom{\lfloor \lg n\rfloor + h - 1}{\lfloor \lg n\rfloor}$ leaves (which is at least $n^{\lg(h/\lg n) - 1}$ if $h \leq n \lg n$, and at least $\left(\frac{n}{2}\right)^{\lg(h/\lg n)}$ in general). Recall that the number of queries used by our algorithms is proportional to widths of the employed trees (cf. Lemmata 3.2 and 3.6), thus by the above it is also quasi-polynomial.

The rest of this section is devoted to the proof of Theorem 5.1. Observe first that Point (2) of this theorem is a direct consequence of Point (1), because $\|\mathcal{F}\|_V = \|\mathcal{F}\|_{C_{n,h},V}$, by Lemma 3.4. It is thus enough to prove Point (1). Moreover, we can assume that $h \geq 1$, because for height $h = 0$ there exists only one tree $U = V = \langle\rangle$, which is $(n, 0)$-universal.

In order to prove Point (1), fix an equitable tree T of height $h \geq 1$ and of width (exactly) n. The core of our proof is a "difficult example": a fixed-point expression $\mathcal{F}_T$ such that if $\|\mathcal{F}_T\|_{U,V}$ is correct (i.e., equal to $\|\mathcal{F}_T\|$) for some U then T embeds in U, under some mild assumptions saying, roughly, that V is nontrivial. In order to achieve this property, we

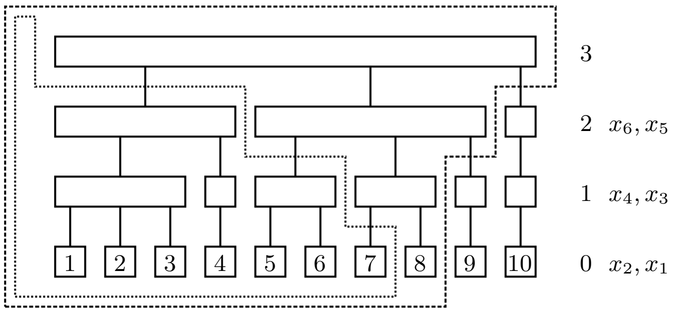


Figure 2 Example equitable tree T of height 3 with numbers in leaves. On the right: layer numbers and corresponding variable names. Dotted and dashed areas represent bits set to 1 in the values $A(v)$ and $B(v)$ for ancestors v of the 8-th leaf. Thus, the 8-th bit of $f_3(x_1, x_2, x_3, x_4, x_5, x_6)$ is 1 if at least the following bits are set to 1: the first 8 bits of x_1, the first 7 bits of x_2, the first 8 bits of x_3, the first 6 bits of x_4, the first 9 bits of x_5, and the first 4 bits of x_6.

construct $\mathcal{F}_T$ that has only one sup-dominion decomposition, which has the shape of the tree T (this claim needs not be proven, but it helps to understand the construction). The construction is to a large degree motivated by the work of Czerwiński *et al.* [8]. Existence of $\mathcal{F}_T$ for every T essentially implies that U is universal, as soon as we deal with the additional assumptions on the tree V, which is done at the very end.

Let us assign numbers $1, \dots, n$ to leaves of T, consecutively from left to right. For every node v we define two values $A(v), B(v) \in \mathbb{B}^n$, as follows:

- if the leftmost leaf descendant of v has number i, then bits number $1, 2, \dots, i-1$ in $A(v)$ are set to 1 (and the remaining bits are set to 0), and
- if the rightmost leaf descendant of v has number j, then bits number $1, 2, \dots, j$ in $B(v)$ are set to 1 (and the remaining bits are set to 0).

Having these values, for every level $\ell \in \{0, \dots, h\}$ we define two functions, $g_\ell^+, g_\ell^- : \mathbb{B}^n \to \mathbb{B}^n$. For every $x \in \mathbb{B}^n$, we consider the rightmost node v at level ℓ such that $A(v) \le x$ (such a v exists, because $A(v) = \mathbf{0}$ for the extreme-leftmost node v), and we take

$$g_\ell^+(x) = B(v), \qquad \text{and} \qquad g_\ell^-(x) = \begin{cases} \mathbf{1} & \text{if } x = \mathbf{1}, \\ A(v) & \text{otherwise.} \end{cases}$$

Finally, for every level $\ell \in \{0, \dots, h\}$ we define a function $f_\ell \colon (\mathbb{B}^n)^{2\ell} \to \mathbb{B}^n$, by induction on ℓ, as follows:

$$\begin{aligned} &f_0() = \mathbf{1}, && \text{and} \\ &f_\ell(x_1, \dots, x_{2\ell-2}, x_{2\ell-1}, x_{2\ell}) = f_{\ell-1}(x_1, \dots, x_{2\ell-2}) * g_{\ell-1}^-(x_{2\ell-1}) * g_{\ell-1}^+(x_{2\ell}) && \text{for } \ell \ge 1. \end{aligned}$$

The sequence of fixed-point operators corresponding to f_ℓ is $\Theta_\ell = \langle \mu, \nu \rangle^\ell$. As the resulting expression we take $\mathcal{F}_T = (\Theta_h, f_h, (\mathbf{0}, \mathbf{1}))$. See Figure 2 for an example.

Directly from the definition it follows that for every node v at level ℓ,

$$g_\ell^+(A(v)) = B(v), \qquad g_\ell^-(B(v)) = B(v), \qquad g_\ell^-(x) \le A(v) \text{ if } x < B(v). \tag{4}$$

Another useful property is that for every fixed-point expression of the form $(\Theta, f * C, (A, B))$ we can move the multiplicative constant C from "the function" to "the bound" (anyway, it will be just multiplied by the function):

$$\|(\Theta, f * C, (A, B))\| = \|(\Theta, f, (A, B * C))\|,$$

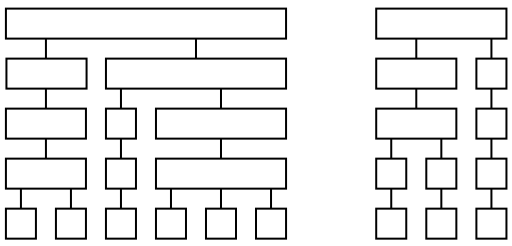


Figure 3 Example tree T (left) and the corresponding comb C_T (right).

or, more specifically (assuming $\ell \geq 1$),

$$\|(\Theta_{\ell-1}, (f_\ell{}^{\upharpoonright D})^{\upharpoonright C}, (A, B))\| = \|(\Theta_{\ell-1}, f_{\ell-1}, (A, B * g^-_{\ell-1}(C) * g^+_{\ell-1}(D)))\|. \tag{5}$$

The same holds for the value parameterised by two trees, $\|\cdot\|_{U,V}$.

Analysing the definition of $\mathcal{F}_T$, it is not difficult to prove by induction on ℓ that

$$\|(\Theta_\ell, f_\ell, (\mathbf{0}, B(v)))\| \geq B(v).$$

for every node v of T located at level ℓ (see Appendix E for details). In particular, taking the root of T as v, we obtain the following lemma.

▶ **Lemma 5.2.** $\|\mathcal{F}_T\| = \mathbf{1}$.

Next, we define trees used on the "inf" side. Namely, for every $\ell \in \{0, \dots, h\}$ we define an equitable tree C_ℓ ("comb") of height ℓ:

- $C_0 = \langle\rangle$,
- $C_\ell = \langle C_{\ell-1}\rangle$ for $\ell \geq 1$, if every node at level $\ell - 1$ in T has at most one child, and
- $C_\ell = \langle C_{\ell-1}, S_{\ell-1}\rangle$ for $\ell \geq 1$, if some node at level $\ell - 1$ in T has at least two children, where $S_{\ell-1}$ is the (unique) tree of height $\ell - 1$ having exactly one leaf.

Moreover, we define $C_T = C_h$. See Figure 3 for an illustration. Notice that the information about T is "shifted by one level" in C_T; in particular C_1 always equals $\langle\langle\rangle\rangle$ (leaves of T have no children), and the shape of C_T does not depend on the fact whether or not the root of T has at least two children. Attention: trees C_ℓ and S_ℓ defined here should not be confused with universal trees $C_{n,h}$ and $S_{n,h}$ defined earlier.

It is easy to see by induction on $\ell \in \{1, \dots, h\}$ that the width of C_ℓ plus the number of nodes of T at level $\ell - 1$ is at most $n + 1$. In particular, the width of C_T is at most n.

For a node v of T we write $T(v)$ for the subtree of T starting at v.

▶ **Lemma 5.3.** *Let U be an equitable tree of height $\ell \in \{0, \dots, h\}$, and let v be a node of T located at level ℓ. If $T(v)$ does not embed in U, then $\|(\Theta_\ell, f_\ell, (A(v), B(v)))\|_{U,C_\ell} < B(v)$. If additionally v has at most one child, then $\|(\Theta_\ell, f_\ell, (A(v), B(v)))\|_{U,C_\ell} \leq A(v)$.*

Proof. The proof is by induction on ℓ. For $\ell = 0$ the node v is a leaf, so surely $T(v)$ (i.e., $\langle\rangle$) embeds in U; there is nothing to prove.

Suppose now that $\ell \geq 1$. Let $\Theta'_\ell = \langle\nu\rangle \cdot \langle\mu, \nu\rangle^{l-1}$ and let $U = \langle U_1, \dots, U_p\rangle$. We first prove that for every child c of v and for every $U' \in \{U_1, \dots, U_p\}$,

$$\|(\Theta'_\ell, f_\ell{}^{\upharpoonright A(c)}, (A(c), B(v)))\|_{U',C_\ell} \leq \begin{cases} B(c) & \text{if } T(c) \text{ embeds in } U', \\ A(c) & \text{otherwise.} \end{cases} \tag{6}$$

Denote $E = \|(\Theta'_\ell, f_\ell{\upharpoonright^{A(c)}}, (A(c), B(v)))\|_{U',C_\ell}$. Let

$$B_1 = \|(\Theta_{\ell-1}, (f_\ell{\upharpoonright^{A(c)}}){\upharpoonright^{B(v)}}, (A(c), B(v)))\|_{U',C_{\ell-1}} \qquad \text{and} \tag{7}$$

$$B_2 = \|(\Theta_{\ell-1}, (f_\ell{\upharpoonright^{A(c)}}){\upharpoonright^{B_1}}, (A(c), B_1))\|_{U',S_{\ell-1}}. \tag{8}$$

By definition, E equals B_1 when $C_\ell = \langle C_{\ell-1}\rangle$, and equals B_2 when $C_\ell = \langle C_{\ell-1}, S_{\ell-1}\rangle$. Because $g^-_{\ell-1}(B(v)) = B(v) \geq B(c)$ and $g^+_{\ell-1}(A(c)) = B(c)$ (cf. Equalities (4)), we can simplify Equalities (7) and (8) (using also Equality (5)) to

$$B_1 = \|(\Theta_{\ell-1}, f_{\ell-1}, (A(c), B(c)))\|_{U',C_{\ell-1}}, \qquad \text{and} \tag{9}$$

$$B_2 = \|(\Theta_{\ell-1}, f_{\ell-1}, (A(c), B(c) * g^-_{\ell-1}(B_1)))\|_{U',S_{\ell-1}}. \tag{10}$$

We now have three cases.

First, suppose that $T(c)$ embeds in U'. In this case $B_2 \leq B_1 \leq B(c)$ by Lemma 4.6 and Equalities (8) and (9), so $E \leq B(c)$ (no matter whether E equals B_1 or B_2).

Next, suppose that $T(c)$ does not embed in U' and that $C_\ell = \langle C_{\ell-1}\rangle$. By definition of C_ℓ this means that c (and every other node at level $\ell - 1$) has at most one child. Thus, the induction hypothesis implies that $\|(\Theta_{\ell-1}, f_{\ell-1}, (A(c), B(c)))\|_{U',C_{\ell-1}} \leq A(c)$, that is, $E = B_1 \leq A(c)$ (by Equality (9)).

Finally, suppose that $T(c)$ does not embed in U', but $C_\ell = \langle C_{\ell-1}, S_{\ell-1}\rangle$. In this case the induction hypothesis implies that $\|(\Theta_{\ell-1}, f_{\ell-1}, (A(c), B(c)))\|_{U',C_{\ell-1}} < B(c)$, that is, $B_1 < B(c)$ (by Equality (9)). Then $g^-_{\ell-1}(B_1) \leq A(c)$ (cf. Equalities (4)), so $E = B_2 \leq A(c)$ by Lemma 4.6 and Equality (10). This finishes the proof of Inequality (6).

Recall now that $\|(\Theta_\ell, f_\ell, (A(v), B(v)))\|_{U,C_\ell}$ is defined as A_p, where $A_j = \|(\Theta'_\ell, f_\ell{\upharpoonright^{A_{j-1}}}, (A_{j-1}, B(v)))\|_{U_j,C_\ell}$ for $j \in \{1, \dots, p\}$ and $A_0 = A(v)$. Notice that A_0 equals $A(c)$ for the leftmost child c of v. If $A_{j-1} \leq A(c)$ for some child c of v, by monotonicity we have $A_j \leq \|(\Theta'_\ell, f_\ell{\upharpoonright^{A(c)}}, (A(c), B(v)))\|_{U_j,C_\ell}$, so we can apply Inequality (6). If $A_{j-1} \leq A(c)$ and $T(c)$ does not embed in U_j, we obtain that $A_j \leq A(c)$ as well. Contrarily, if $A_{j-1} \leq A(c)$ and $T(c)$ embeds in U_j, we obtain that $A_j \leq B(c) = A(c')$, where c' is the right sibling of c. We know that $T(v)$ does not embed in U, so $\|(\Theta_\ell, f_\ell, (A(v), B(v)))\|_{U,C_\ell} = A_p \leq A(c)$ for some child c of v (this is the first child that "could not be embedded"), and $A(c) < B(v)$, as required in the thesis. If moreover v has exactly one child (i.e., c is the only child of v), then $A(c) = A(v)$. ◀

▶ **Corollary 5.4.** *Let U, V be equitable trees of height h. If $\|\mathcal{F}_T\|_{U,V} = \|\mathcal{F}_T\|$ and C_T embeds in V, then T embeds in U. If $\|\widetilde{\mathcal{F}_T}\|_{U,V} = \|\widetilde{\mathcal{F}_T}\|$ and C_T embeds in U, then T embeds in V.*

Proof. For the first part, by Lemmata 5.2 and 4.8 we have $\mathbf{1} = \|\mathcal{F}_T\| = \|\mathcal{F}_T\|_{U,V} \leq \|\mathcal{F}_T\|_{U,C_T}$, that is, $\|\mathcal{F}_T\|_{U,C_T} = \mathbf{1}$. But Lemma 5.3 (where as v we take the root of T) allows $\|\mathcal{F}_T\|_{U,C_T}$ to be $\mathbf{1}$ only if T embeds in U. The second part is a consequence of the first part and of the equalities $\|\widetilde{\mathcal{F}_T}\| = \overline{\|\mathcal{F}_T\|}$ (Proposition 2.4) and $\|\widetilde{\mathcal{F}_T}\|_{U,V} = \overline{\|\mathcal{F}_T\|_{V,U}}$ (Lemma 4.7). ◀

We now finish the proof of (Point (1) of) Theorem 5.1. From this point, the tree T is no longer fixed. Recall that we have two equitable trees U, V of height $h \geq 1$, and we assume that $\|\mathcal{F}\|_{U,V} = \|\mathcal{F}\|$ for all fixed-point expressions of the form $\mathcal{F} = (\Theta, f, (\mathbf{0}, \mathbf{1}))$, where $|\Theta|_\mu = |\Theta|_\nu = h$. The goal is to prove that U and V are (n, h)-universal, that is, every equitable tree T of height h and width at most n can be embedded in U and V.

The difficulty is that in order to prove that a tree T embeds in U, we already need to know that some tree, namely C_T, embeds in V, and in order to prove that a tree T embeds in V, we already need to know that some tree, namely C_T, embeds in U (cf. Corollary 5.4). In order to deal with this circular dependency, we argue that C_T is "simpler" than T. To compare these trees, we introduce a parameter called a stick length.

The *stick length* of an equitable tree T of height h is the greatest number $s \in \{0, \dots, h\}$ such that all nodes at levels $0, 1, \dots, s$ have at most one child. We prove by induction on $h - s$ that every equitable tree T of height h, width at most n, and stick length s embeds in U and in V.

The only tree T of height h and stick length $s = h$ is S_h (a single branch); it clearly embeds in every tree of height h, in particular in U and in V.

For an induction step, consider an equitable tree T of height h, width at most n, and stick length $s < h$. Recall that the corresponding tree C_T has height h and width at most n. It is also easy to see that it has stick length $s + 1$ (if all nodes at some level ℓ in T have at most one child, then all nodes at level $\ell + 1$ in C_T have at most one child). Thus, C_T embeds in U and in V by the induction hypothesis. If T has width n, it embeds in U and in V by Corollary 5.4, as required. If the width is smaller, we consider the tree T' of width n obtained from T by attaching an appropriate number of trees S_{h-1} directly below the root. This tree also has stick length s, height h, and is equitable, so it embeds in U and in V by the above, and hence T embeds as well.

▶ Remark 5.5. Theorem 5.1 can be strengthened in the following way: it is enough to assume that the equalities $\|\mathcal{F}\|_{U,V} = \|\mathcal{F}\|$ and $\|\mathcal{F}\|_U = \|\mathcal{F}\|$ hold for those monotonic functions f, where every output bit is a logical OR or a logical AND of some input bits. Indeed, all functions constructed in the proof are of this kind. This makes a connection with parity games (cf. Introduction).

6 Conclusion

It is well known that a nested fixed point like $\mu x.\nu y.\mu x.f(x, y, z)$ can be computed by a formula

$$\mu^\gamma z.\nu^\beta y.\mu^\alpha x.f(x, y, z),$$

for sufficiently large ordinals α, β, γ, where $\theta^\eta x$ means that only η iterations of a fixed point θx are computed (cf. e.g. [27]). Our algorithms refine this idea in two ways: the iterations follow a more subtle tree structure, and moreover they are combined with narrowing the scope according to the nested $A + B * f$ pattern. In the asymmetric case, the restrictions apply only to one kind of fixed points, but the complexity – somewhat surprisingly – actually improves as we eventually compute a single fixed point (albeit of a larger system).

Our black-box algorithms, when adapted to parity games, match the best complexity known so far, but do not improve it. Our lower bound, in view of Remark 5.5, gives an evidence that in the algorithm by Jurdziński and Morvan [19] the use of (not others but) universal trees is indeed necessary.

This may suggest that any polynomial algorithm for parity games – if it exists – should involve some other structure of the game, which remains to be discovered. On the other hand, it is an intriguing question if the lower bound can be strengthen to an "absolute" lower bound for the number of queries, like the aforementioned (almost) quadratic lower bound [24]. A recently developed theory around the complexity of computing the Nash equilibria [9] warns us that some problems in NP ∩ co-NP may be hard.

References

1 André Arnold and Damian Niwiński. *Rudiments of μ-Calculus*. Studies in Logic and the Foundations of Mathematics. Elsevier, 2001.

2 Julien Bernet, David Janin, and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO Theor. Informatics Appl.*, 36(3):261–275, 2002. doi:10.1051/ita:2002013.

3 Mikołaj Bojańczyk and Wojciech Czerwiński. *An automata toolbox*. Informal lecture notes, 2018. URL: `https://www.mimuw.edu.pl/~bojan/upload/reduced-may-25.pdf`.

4 Anca Browne, Edmund M. Clarke, Somesh Jha, David E. Long, and Wilfredo R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theor. Comput. Sci.*, 178(1-2):237–255, 1997. `doi:10.1016/S0304-3975(96)00228-9`.

5 Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 252–263. ACM, 2017. `doi:10.1145/3055399.3055409`.

6 Krishnendu Chatterjee, Wolfgang Dvořák, Monika Henzinger, and Alexander Svozil. Quasipolynomial set-based symbolic algorithms for parity games. In Gilles Barthe, Geoff Sutcliffe, and Margus Veanes, editors, *LPAR-22. 22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning, Awassa, Ethiopia, 16-21 November 2018*, volume 57 of *EPiC Series in Computing*, pages 233–253. EasyChair, 2018. URL: `http://www.easychair.org/publications/paper/L8b1`.

7 Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata: New tools for infinite duration games. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2019. `doi:10.1007/978-3-030-17127-8_1`.

8 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 Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2333–2349. SIAM, 2019. `doi:10.1137/1.9781611975482.142`.

9 Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *Commun. ACM*, 52(2):89–97, 2009. `doi:10.1145/1461928.1461951`.

10 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377. IEEE Computer Society, 1991. `doi:10.1109/SFCS.1991.185392`.

11 E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comput. Sci.*, 258(1-2):491–522, 2001. `doi:10.1016/S0304-3975(00)00034-7`.

12 John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In Hakan Erdogmus and Klaus Havelund, editors, *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software, Santa Barbara, CA, USA, July 10-14, 2017*, pages 112–121. ACM, 2017. `doi:10.1145/3092282.3092286`.

13 Hugo Gimbert and Rasmus Ibsen-Jensen. A short proof of correctness of the quasi-polynomial time algorithm for parity games. *CoRR*, abs/1702.01953, 2017. `arXiv:1702.01953`.

14 Daniel Hausmann and Lutz Schröder. Computing nested fixpoints in quasipolynomial time. *CoRR*, abs/1907.07020v2, 2019. `arXiv:1907.07020v2`.

15 Daniel Hausmann and Lutz Schröder. Quasipolynomial computation of nested fixpoints. *CoRR*, abs/1907.07020v3, 2020. `arXiv:1907.07020v3`.

16 Marcin Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998. `doi:10.1016/S0020-0190(98)00150-1`.

17 Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Lille, France, February 2000, Proceedings*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000. `doi:10.1007/3-540-46541-3_24`.

18 Marcin Jurdziński and Ranko Lazić. Succinct progress measures for solving parity games. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–9. IEEE Computer Society, 2017. `doi:10.1109/LICS.2017.8005092`.

19 Marcin Jurdziński and Rémi Morvan. A universal attractor decomposition algorithm for parity games. *CoRR*, abs/2001.04333, 2020. `arXiv:2001.04333`.

20 Marcin Jurdziński, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 117–123. ACM Press, 2006. URL: `http://dl.acm.org/citation.cfm?id=1109557.1109571`.

21 Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 639–648. ACM, 2018. `doi:10.1145/3209108.3209115`.

22 Karoliina Lehtinen, Sven Schewe, and Dominik Wojtczak. Improving the complexity of Parys' recursive algorithm. *CoRR*, abs/1904.11810, 2019. `arXiv:1904.11810`.

23 Robert McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993. `doi:10.1016/0168-0072(93)90036-D`.

24 Paweł Parys. Some results on complexity of μ-calculus evaluation in the black-box model. *RAIRO - Theor. Inf. and Applic.*, 47(1):97–109, 2013. `doi:10.1051/ita/2012030`.

25 Paweł Parys. Parity games: Zielonka's algorithm in quasi-polynomial time. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 10:1–10:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. `doi:10.4230/LIPIcs.MFCS.2019.10`.

26 Helmut Seidl. Fast and simple nested fixpoints. *Inf. Process. Lett.*, 59(6):303–308, 1996. `doi:10.1016/0020-0190(96)00130-5`.

27 Igor Walukiewicz. Monadic second order logic on tree-like structures. In Claude Puech and Rüdiger Reischuk, editors, *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, volume 1046 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 1996. `doi:10.1007/3-540-60922-9_33`.

28 Shipei Zhang, Oleg Sokolsky, and Scott A. Smolka. On the parallel complexity of model checking in the modal mu-calculus. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*, pages 154–163. IEEE Computer Society, 1994. `doi:10.1109/LICS.1994.316075`.

29 Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. `doi:10.1016/S0304-3975(98)00009-7`.

A Proof of Proposition 2.3

▶ **Proposition 2.3.** *If $A \leq C \leq \|(\Theta, f, (A, B))\| \leq D \leq B$, then*

$$\|(\Theta, f, (A, B))\| = \|(\Theta, f, (C, D))\|.$$

Proof. The proof is by induction on the length of Θ.

In the base case of $\Theta = \langle\rangle$, the thesis amounts to $A + B * f() = C + D * f()$, assuming that $A \leq C \leq A + B * f() \leq D \leq B$. But we have that $B * f() \leq A + B * f() \leq D$ and

$B * f() \leq f()$, which implies that $B * f() \leq D * f()$. Since moreover $A \leq C$, we obtain that $A + B * f() \leq C + D * f()$. The converse inequality $A + B * f() \geq C + D * f()$ can be proven analogously.

Suppose now that $\Theta = \langle\theta\rangle \cdot \Theta'$. Let $E = \|(\Theta, f, (A, B))\|$. Then $E = \theta x.g(x)$ where $g(x) = \|(\Theta', f^{\upharpoonright x}, (A, B))\|$. By Proposition 2.2 it follows that $A \leq g(x) \leq B$ for all $x \in \mathbb{B}^n$, which due to the inequalities $A \leq C$ and $D \leq B$ implies that $A \leq C * g(x) \leq g(x) \leq D + g(x) \leq B$. By the induction hypothesis,

$$g(x) = \|(\Theta', f^{\upharpoonright x}, (A, B))\| = \|(\Theta', f^{\upharpoonright x}, (C * g(x), D + g(x)))\|.$$

Let $h(x, y) = \|(\Theta', f^{\upharpoonright x}, (C * g(y), D + g(y)))\|$, so that $h(x, x) = g(x)$. By Equality (3), $E = \theta x.g(x) = \theta y.\theta x.h(x, y)$, so $E = \theta x.h(x, E)$. Recalling that $g(E) = E$, by assumption we have that $C \leq g(E) \leq D$. Unravelling the definition of h in $E = \theta x.h(x, E)$, we obtain that

$$E = \theta x.\|(\Theta', f^{\upharpoonright x}, (C * g(E), D + g(E)))\| = \theta x.\|(\Theta', f^{\upharpoonright x}, (C, D))\| = \|(\Theta, f, (C, D))\|,$$

as required. ◀

B Proof of Lemma 3.4

Heading towards a proof of Lemma 3.4, we first note an analogue to Proposition 2.2 and Lemma 4.6, which follows by a straightforward induction:

▶ **Lemma B.1.** *For every fixed-point expression* $(\Theta, f, (A, B))$ *and for every equitable tree* V *of height* $|\Theta|_\nu$,

$$A \leq \|(\Theta, f, (A, B))\|_V \leq B.$$

As already mentioned, while proving Lemma 3.4 we make use of the following simple fact.

▶ **Proposition B.2.** *For a monotone mapping* $f\colon \mathbb{B}^n \to \mathbb{B}^n$,

$$\mu x.f(x) = \underbrace{f(\dots(f}_{n}(\mathbf{0}))\dots).$$

We now generalize Lemma 3.4 a bit, to a variant suitable for an inductive proof (Lemma 3.4 follows from Lemma B.3 by taking $A' = A$):

▶ **Lemma B.3.** *Let* $(\Theta, f, (A, B))$ *be a fixed-point expression, and let* V *be an equitable tree of height* $|\Theta|_\nu$. *If* $A \leq A' \leq \|(\Theta, f, (A, B))\|_V$, *then*

$$\|(\Theta, f, (A, B))\|_V = \|(\Theta, f, (A', B))\|_{C_{n,|\Theta|_\mu}, V}.$$

Proof. Induction on the length of Θ. Let $h = |\Theta|_\mu$. For $\Theta = \langle\rangle$ on the one hand due to $A \leq A'$ we have

$$\|(\Theta, f, (A, B))\|_V = A + B * f() \leq A' + B * f() = \|(\Theta, f, (A', B))\|_{C_{n,h}, V},$$

and on the other hand due to $A' \leq \|(\Theta, f, (A, B))\|_V$ we have

$$A' + B * f() \leq A + B * f() + B * f() = A + B * f().$$

Next, suppose that $\Theta = \langle\nu\rangle \cdot \Theta'$ and $V = \langle V_1, \dots, V_p\rangle$. Let $B_0 = B'_0 = B$ and for $j \in \{1, \dots, p\}$, let

$$B_j = \|(\Theta', f^{\upharpoonright B_{j-1}}, (A, B_{j-1}))\|_{V_j}, \qquad B'_j = \|(\Theta', f^{\upharpoonright B'_{j-1}}, (A', B'_{j-1}))\|_{C_{n,h}, V_j}.$$

Since $A \le A' \le \|(\Theta, f, (A, B))\|_V = B_p \le B_{p-1} \le \dots \le B_0$ (the inequalities between the B_j's are by Lemma B.1), we have by the induction hypothesis

$$B_j = \|(\Theta', f^{\upharpoonright B_{j-1}}, (A', B_{j-1}))\|_{C_{n,h}, V_j}.$$

Thus, $B_{j-1} = B'_{j-1}$ implies $B_j = B'_j$, and since $B_0 = B'_0 = B$, we have $\|(\Theta, f, (A, B))\|_V = B_p = B'_p = \|(\Theta, f, (A', B))\|_{C_{n,h}, V}$.

Finally, suppose that $\Theta = \langle\mu\rangle \cdot \Theta'$. Let us define

$$E_0 = \mathbf{0}, \quad E_j = \|(\Theta', f^{\upharpoonright E_{j-1}}, (A, B))\|_V \quad \text{for } j \in \{1, \dots, n\}, \quad \text{and} \tag{11}$$

$$A'_0 = A', \quad A'_j = \|(\Theta', f^{\upharpoonright A'_{j-1}}, (A'_{j-1}, B))\|_{C_{n,h-1}, V} \quad \text{for } j \in \{1, \dots, n\}. \tag{12}$$

Let $E = E_n$. By Proposition B.2 we have $E = \mu x.\|(\Theta', f^{\upharpoonright x}, (A, B))\|_V = \|(\Theta, f, (A, B))\|_V$. Simultaneously, by the definition of $\|(\Theta, f, (A', B))\|_{C_{n,h}, V}$ we have $\|(\Theta, f, (A', B))\|_{C_{n,h}, V} = A'_n$. Moreover, by Lemma 4.6 we have $A \le A' = A'_0 \le A'_1 \le \dots \le A'_n$, that is, $A \le A'_j$ for all $j \in \{0, \dots, n\}$. We now prove by induction on $j \in \{0, \dots, n\}$ that $E_j \le A'_j \le E$ (for $j = n$ we obtain the required equality $E = E_n = A'_n$). For $j = 0$ the inequality boils down to $\mathbf{0} \le A' \le E$, which holds by assumption. Suppose thus that $j \in \{1, \dots, n\}$ and that $E_{j-1} \le A'_{j-1} \le E$. By monotonicity and by the induction hypothesis it follows that

$$E_j \overset{(11)}{\le} \|(\Theta', f^{\upharpoonright A'_{j-1}}, (A'_{j-1}, B))\|_V = \|(\Theta', f^{\upharpoonright A'_{j-1}}, (A'_{j-1}, B))\|_{C_{n,h}, V} = A'_j.$$

Again by monotonicity and again by the induction hypothesis, since $A \le A'_{j-1} \le E = \|(\Theta', f^{\upharpoonright E}, (A, B))\|_V$,

$$A'_j \overset{(12)}{\le} \|(\Theta', f^{\upharpoonright E}, (A'_{j-1}, B))\|_{C_{n,h}, V} = \|(\Theta', f^{\upharpoonright E}, (A, B))\|_V = E. \qquad \blacktriangleleft$$

C An explicit definition of the system of equations

In this section we give an explicit description of the system of equations generated by Algorithm 1 for $\|(\Theta, f, (A, B))\|_V$, where $|\Theta|_\nu = h$. Without loss of generality we may assume that $\Theta = \langle\mu\rangle \cdot \langle\nu, \mu\rangle^h$; to reach such a situation from a general case, one can add additional parameters (quantified by μ) on which f does not depend (and compress neighbouring occurrences of μ using Equality (3)). Suppose that leaves of V are numbered $1, \dots, m$ from left to right (where $m = |V|$). For all $i \in \{1, \dots, m\}$ and $\ell \in \{0, \dots, h\}$, let $\lambda^\ell(i)$ be the least number $j \in \{0, \dots, m-1\}$ such that leaves number $j + 1$ and i have the same ancestor at level ℓ, and let $\rho^\ell(i)$ be the greatest number $j \in \{1, \dots, m\}$ such that leaves number i and j have the same ancestor at level ℓ (in particular, $\lambda^0(i) = i - 1$ and $\rho^0(i) = i$). In other words, if v is the ancestor of the i-th leaf located at level ℓ, then $\lambda^\ell(i)$ is the number of the leftmost leaf descendant of v, decreased by one, and $\rho^\ell(i)$ is the number of the rightmost leaf descendant of v. In the system, we use variables $\mathsf{x}_1, \dots, \mathsf{x}_m$ and additionally the variable x_0 that is valuated to $\mathbf{1}$. For every $i \in \{1, \dots, m\}$ we have an equation

$$\mathsf{x}_i = \mathsf{x}_{\lambda^0(i)} * \mathsf{f}(\mathsf{x}_{\rho^0(i)}, \mathsf{x}_{\lambda^0(i)}, \mathsf{x}_{\rho^1(i)}, \mathsf{x}_{\lambda^1(i)}, \mathsf{x}_{\rho^2(i)}, \mathsf{x}_{\lambda^2(i)}, \dots, \mathsf{x}_{\rho^{h-1}(i)}, \mathsf{x}_{\lambda^{h-1}(i)}, \mathsf{x}_{\rho^h(i)}). \tag{$\star$}$$

It is the variable x_m that stores the result (i.e., $x_{res} = \mathsf{x}_m$).

We now prove that the system generated by Algorithm 1 is indeed of the above form. Originally, the last argument of the procedure Generate is a subtree of the input tree V. Let us consider a modification of this procedure, where the last argument is a node of V, being the root of this subtree (and then, in the procedure, we consider the subtree starting in this node).

Consider now a recursive call

$$\textsc{Generate}(x, \langle\mu\rangle \cdot \langle\nu,\mu\rangle^\ell, \mathbf{f}, \mathbf{B}, u) \qquad \text{with } \mathbf{f} = \mathsf{f}(?, \dots, ?, y_{2\ell+2}, \dots, y_{2h+1})$$

performed during the considered execution of Algorithm 1. It is easy to prove (by induction on the depth of the recursion) that, if i is the number of the rightmost leaf descendant of the node u,

- u is located at level ℓ,
- for every $j \in \{\ell+1, \dots, h\}$, the argument y_{2j} contains the variable $\mathsf{x}_{\lambda^{j-1}(i)}$, and the argument y_{2j+1} contains the variable $\mathsf{x}_{\rho^j(i)}$,
- the argument $\mathbf{B}$ contains the variable $\mathsf{x}_{\lambda^\ell(i)}$, and
- the argument x contains the variable $\mathsf{x}_{\rho^\ell(i)}$, that is, x_i.

From the above claim it follows that equations generated in line 4 of the algorithm are indeed of the form ($\star$).

D Proof of Lemma 3.5

▶ **Lemma 3.5.** *Let $\mathcal{F} = (\Theta, f, (\mathbf{0}, \mathbf{1}))$ be a fixed-point expression, and let V be an equitable tree of height $|\Theta|_\nu$. The value assigned to the variable x_{res} in the least solution of the system of equations generated by Algorithm 1 equals $\|\mathcal{F}\|_V$.*

Proof. In order to facilitate an inductive proof of this lemma, we slightly modify Algorithm 1. Namely, we replace line 14 by the following three lines:

```
$x' = \textsc{FreshVariable}()$;
$\textsc{Generate}(x, \Theta', \mathbf{f}^{\upharpoonright x'}, \mathbf{B}, V)$;
output "$x' = x$";
```

This way, we replace some occurrences of the variable x in the resulting system of equations by a fresh variable x', and we add an equation ensuring that x' equals x. It should be clear that there is a one-to-one correspondence between solutions of the original system of equations and solutions of the new one; in particular the least solution is the same (modulo the difference that in the new system some variables are multiplicated). Thus, from now on we only consider the modified algorithm.

Let d be the length of Θ. We prove a claim concerning any recursive call

$$\textsc{Generate}(x, \widehat{\Theta}, \mathbf{f}, \mathbf{B}, \widehat{V}) \qquad \text{with } \mathbf{f} = \mathsf{f}(?, \dots, ?, y_{k+1}, \dots, y_d).$$

We assume that k (i.e., the number of ?'s) equals the length of $\widehat{\Theta}$, and $\widehat{V}$ is an equitable tree of height $|\widehat{\Theta}|_\nu$ (we use here symbols $\widehat{\Theta}$ and $\widehat{V}$, not Θ and V, in order to distinguish these arguments from the original sequence Θ and from the original tree V). Moreover, we assume that the variable contained in the argument x is not contained in any of the arguments $y_{k+1}, \dots, y_d, \mathbf{B}$. All calls appearing in the modified version of the algorithm satisfy the above two conditions (while the second condition is not satisfied by the original algorithm; this is why we consider the modification). Let $\mathcal{S}$ be the system of equations generated by the above call. A *solution* of $\mathcal{S}$ is a function that maps every variable appearing in $\mathcal{S}$ (including those that do not appear on the left side of any equation) to an element of $\mathbb{B}^n$. Consider also a valuation $val\colon \{y_{k+1},\dots,y_d,\mathbf{B}\}\to\mathbb{B}^n$. Based on the the original function f (of arity d), the valuation induces a new function f_{val} of arity k, in an obvious way. Let $B = val(\mathbf{B})$ and $E = \|(\widehat{\Theta}, f_{val}, (\mathbf{0},B))\|_{\widehat{V}}$.

▷ Claim.

(1) For every solution R of $\mathcal{S}$ that extends val it holds $R(x)\ge E$, and

(2) there exists a solution R of $\mathcal{S}$ that extends val and such that $R(x)=E$.

Equivalently, the claim says that in the least solution of $\mathcal{S}$ among those that extend the valuation val, the variable x is valuated to E. In particular, while considering the main call of the algorithm (i.e., the call in line 17), this claim gives us the thesis of the lemma. It thus remains to prove the claim, which is done by induction on k.

For $k=0$ (i.e., $\widehat{\Theta}=\langle\rangle$) the system $\mathcal{S}$ consists of a single equation, namely $x = \mathbf{B}*\mathsf{f}(y_1,\dots,y_d)$. The valuation val already assigns values to the variables $y_1,\dots,y_d,\mathbf{B}$ (but not to x), so there is a unique solution that extends val; it maps x to $B*f_{val}()$, that is, to E.

Next, suppose that $\widehat{\Theta}=\langle\nu\rangle\cdot\widehat{\Theta}'$. Let $\widehat{V}=\langle\widehat{V}_1,\dots,\widehat{V}_p\rangle$, let $B_0=B$, and for $j\in\{1,\dots,p\}$ let $B_j=\|(\widehat{\Theta}',(f_{val})^{\upharpoonright B_{j-1}},(\mathbf{0},B_{j-1}))\|_{\widehat{V}_j}$. Then, by definition, $E=B_p$. Moreover, for every $j\in\{1,\dots,p\}$, let $\mathcal{S}_j$ be the system of equations generated by the call $\textsc{Generate}(\mathbf{B}_j,\widehat{\Theta}',\mathbf{f}^{\upharpoonright\mathbf{B}_{j-1}},\mathbf{B}_{j-1},\widehat{V}_j)$ in line 11 of the algorithm.

Consider now a solution R of $\mathcal{S}$ that extends val. We prove by induction on $j\in\{0,\dots,p\}$ that $R(\mathbf{B}_j)\ge B_j$; for $j=p$ this gives Point (1) of the claim, because $\mathbf{B}_p=x$ and $B_p=E$. For $j=0$ we have $\mathbf{B}_0=\mathbf{B}$, so $R(\mathbf{B}_0)=val(\mathbf{B})=B=B_0$. For the induction step, we prove $R(\mathbf{B}_j)\ge B_j$ assuming $R(\mathbf{B}_{j-1})\ge B_{j-1}$, where $j\in\{1,\dots,p\}$. Because $\mathcal{S}_j$ is a part of $\mathcal{S}$, our solution R (when restricted to variables appearing in $\mathcal{S}_j$) is also a solution of $\mathcal{S}_j$. Moreover, it extends $val[\mathbf{B}_{j-1}\mapsto R(\mathbf{B}_{j-1})]$. Observe that the function of arity $k-1$ induced by the valuation $val[\mathbf{B}_{j-1}\mapsto R(\mathbf{B}_{j-1})]$ (i.e., $f_{val[\mathbf{B}_{j-1}\mapsto R(\mathbf{B}_{j-1})]}$) equals $(f_{val})^{\upharpoonright R(\mathbf{B}_{j-1})}$ (below, similar equalities are used implicitly). Thus, by the induction hypothesis (used for this valuation) and by monotonicity,

$$R(\mathbf{B}_j)\ge\|(\widehat{\Theta}',(f_{val})^{\upharpoonright R(\mathbf{B}_{j-1})},(\mathbf{0},R(\mathbf{B}_{j-1})))\|_{\widehat{V}_j}\ge\|(\widehat{\Theta}',(f_{val})^{\upharpoonright B_{j-1}},(\mathbf{0},B_{j-1}))\|_{\widehat{V}_j}=B_j.$$

In order to prove Point (2) of the claim, for every $j\in\{1,\dots,p\}$ we create by the induction hypothesis a solution R_j to $\mathcal{S}_j$ that extends $val[\mathbf{B}_{j-1}\mapsto B_{j-1}]$ and such that $R_j(\mathbf{B}_j)=B_j$. Observe that the systems $\mathcal{S}_1,\dots,\mathcal{S}_p$ have pairwise disjoint sets of variables, except for the variables $y_{k+1},\dots,y_d$ whose values are fixed by val, and except for variables $\mathbf{B}_1,\dots,\mathbf{B}_{p-1}$ shared by consecutive systems, for which we also fix values in a consistent way. It follows that the solutions $R_1,\dots,R_p$ may be merged into a solution R of the whole system $\mathcal{S}$ (being a union of $\mathcal{S}_1,\dots,\mathcal{S}_p$). This solution extends val, and satisfies $R(x)=R_p(\mathbf{B}_p)=B_p=E$.

We now come to the last case, namely $\widehat{\Theta}=\langle\mu\rangle\cdot\widehat{\Theta}'$. Let $\mathcal{S}'$ be the system of equations generated by the call $\textsc{Generate}(x,\widehat{\Theta}',\mathbf{f}^{\upharpoonright x'},\mathbf{B},\widehat{V})$ in the modified line 14 of the algorithm; in other words, this is $\mathcal{S}$ without the equation $x'=x$. Recall that, in this case, E is the least fixed point of the mapping $C\mapsto\|(\widehat{\Theta}',(f_{val})^{\upharpoonright C},(\mathbf{0},B))\|_{\widehat{V}}$. For Point (1) of the claim, consider a solution R of $\mathcal{S}$ that extends val. This is also a solution of $\mathcal{S}'$, and $R(x')=R(x)$. Thus, by the induction hypothesis used for the valuation $val[x'\mapsto R(x)]$,

$$R(x)\ge\|(\widehat{\Theta}',(f_{val})^{\upharpoonright R(x)},(\mathbf{0},B))\|_{\widehat{V}}.$$

By Equalities (2) this means that $R(x)$ can only be greater than the least fixed point of the aforementioned mapping, that is $R(x)\ge E$. This proves Point (1). For Point (2), the induction hypothesis gives us a solution R of $\mathcal{S}'$ that extends $val[x' \mapsto E]$ and such that $R(x) = \|(\widehat{\Theta}', (f_{val})^{\upharpoonright E}, (\mathbf{0}, B))\|_{\widehat{V}}$, that is, $R(x) = E$. We also have $R(x') = E$, so R is a solution of $\mathcal{S}$ as well. This finishes the proof of the claim, and thus of the whole lemma. ◀

Lemmata 4.1 and 4.2 are immediate consequences of definitions and of Proposition 2.3.

E Proof of Lemma 5.2

As already mentioned, we obtain Lemma 5.2 by taking the root of T as v in the following lemma.

▶ **Lemma E.1.** *For every node v of T located at level ℓ,*

$$\|(\Theta_\ell, f_\ell, (\mathbf{0}, B(v)))\| \geq B(v).$$

Proof. The proof is by induction on ℓ. Let $E = \|(\Theta_\ell, f_\ell, (\mathbf{0}, B(v)))\|$. For $\ell = 0$ we simply have $E = \mathbf{0} + B(v) * \mathbf{1} = B(v)$.

Suppose now that $\ell \geq 1$. Let $\Theta'_\ell = \langle \nu \rangle \cdot \langle \mu, \nu \rangle^{l-1}$, and let c be the rightmost node at level $\ell - 1$ such that $A(c) \leq E$. We first prove that

$$\|(\Theta'_\ell, {f_\ell}^{\upharpoonright E}, (\mathbf{0}, B(c)))\| \geq B(c). \tag{13}$$

To this end, observe that

$$\begin{aligned}\|(\Theta_{\ell-1}, ({f_\ell}^{\upharpoonright E})^{\upharpoonright B(c)}, (\mathbf{0}, B(c)))\| &\overset{(5)}{=} \|(\Theta_{\ell-1}, f_{\ell-1}, (\mathbf{0}, B(c) * g^-_{\ell-1}(B(c)) * g^+_{\ell-1}(E)))\| \\ &\overset{(4)}{=} \|(\Theta_{\ell-1}, f_{\ell-1}, (\mathbf{0}, B(c) * B(c) * B(c)))\| \geq B(c),\end{aligned}$$

where the last inequality is by the induction hypothesis. Moreover, by Proposition 2.2 we have a converse inequality. Together, this means that $B(c)$ is a fixed point of the mapping $x \mapsto \|(\Theta_{\ell-1}, ({f_\ell}^{\upharpoonright E})^{\upharpoonright x}, (\mathbf{0}, B(c)))\|$. By definition $\|(\Theta'_\ell, {f_\ell}^{\upharpoonright E}, (\mathbf{0}, B(c)))\|$ is the greatest fixed points of this mapping, so it can be only greater. This finishes the proof of Inequality (13).

If $B(v) \leq A(c)$, the thesis (i.e., $E \geq B(v)$) is true, since $A(c) \leq E$. Otherwise, $B(v) > A(c)$ implies that $B(v) \geq B(c)$ (because c is not higher in the tree than v). Thus, because E is a fixed point and by monotonicity we have that

$$E = \|(\Theta'_\ell, {f_\ell}^{\upharpoonright E}, (\mathbf{0}, B(v)))\| \geq \|(\Theta'_\ell, {f_\ell}^{\upharpoonright E}, (\mathbf{0}, B(c)))\| \overset{(13)}{\geq} B(c).$$

If c is the rightmost node at level $\ell - 1$, we have $E \geq B(c) = \mathbf{1} \geq B(v)$ and again we are done. In the remaining case, the node c' one to the right from c satisfies $A(c') = B(c) \leq E$, contrary to the definition of c. ◀