On the Way to Alternating Weak Automata

Udi Boker[1]
IDC Herzliya, Israel

Karoliina Lehtinen[2]
Kiel University, Kiel, Germany

Abstract

Different types of automata over words and trees offer different trade-offs between expressivity, conciseness, and the complexity of decision procedures. *Alternating weak automata* enjoy simple algorithms for emptiness and membership checks, which makes transformations into automata of this type particularly interesting. For instance, an algorithm for solving two-player infinite games can be viewed as a special case of such a transformation. However, our understanding of the worst-case size blow-up that these transformations can incur is rather poor. This paper establishes two new results, one on word automata and one on tree automata. We show that:

- Alternating parity word automata can be turned into alternating weak automata of quasi-polynomial (rather than exponential) size.
- Universal co-Büchi tree automata, a special case of alternating parity tree automata, can be exponentially more concise than alternating weak automata.

Along the way, we present a family of game languages, strict for the levels of the weak hierarchy of tree automata, which corresponds to a weak version of the canonical game languages known to be strict for the Mostowski–Rabin index hierarchy.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Alternating automata, Parity games, Parity automata, Weak automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.21

1 Introduction

The interplay between games and automata has been proven fruitful for both game- and automata-theory. In particular, solving two-player infinite games of some winning condition, such as Büchi or parity, reduces to transforming an alternating word automaton with the same acceptance condition into an alternating weak automaton: i) solving a game over an arena A is the same as deciding whether A, seen as a one-letter alternating automaton, is empty; and ii) deciding the emptiness of one-letter alternating weak automata can be done in linear time. The simplicity of weak automata stems from their defining property, the lack of cycles with both accepting and rejecting states.

As a result, the time complexity of solving games is intimately connected to the size blow-up of translating alternating automata to alternating weak automata. Note, however, that the automata-translation question is more general, since automata need not be defined over a one-letter alphabet, and often a binary, or larger, alphabet adds substantial complexity.

Nevertheless, until recently, the best known algorithms for the two problems, with respect to the Büchi and parity conditions, were the same. For Büchi, they involved a quadratic time

[1] Research supported by the Israel Science Foundation grant 1373/16.
[2] Research supported by BMBF project no. 01IS160253 "ARAMiS II".

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).
Editors: Sumit Ganguly and Paritosh Pandya; Article No. 21; pp. 21:1–21:22

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

or size blow-up, and for parity an exponential one [13, 12]. Moreover, some competitive tools for solving parity games are based on the translation from parity to weak automata [23].

About a year ago the picture changed; After many years of incremental progress, two quasi-polynomial algorithms using different techniques for solving parity games were published [5, 8]. So far this breakthrough has not extended to automata-translation.

Recently, a third quasi-polynomial algorithm for solving parity games was published, employing yet another technique [14]. We extend this algorithm for addressing the more general problem, and provide a translation of alternating parity word automata into weak automata that involves only a quasi-polynomial size blow-up.

K. Lehtinen defines in [14] *register games*, which are a parameterised variant of parity games, and bases the new algorithm on them. A central result in the complexity analysis of this algorithm is that Eve wins a parity game with n positions if and only if she wins the corresponding k-register game for every $k > \log n$. The least such k is called the *register-index* of the game. We extend this result, showing that the register-index is also logarithmic in a more refined measure of game size: the maximal number of distinct strongly connected components in it. We call this measure the *ssc-size* of the game.

We link the automata setting to the game setting by defining for every alternating parity word automaton $\mathcal{A}$ and positive integer k, a parameterised alternating parity word automaton $\mathcal{A}_k$, such that $\mathcal{A}_k$ accepts an ultimately periodic word w if and only if Eve wins the k-register game on the arena of the model-checking game over $\mathcal{A}$ and w. When $\mathcal{A}$ has n states, $\mathcal{A}_k$ has $kn^{O(k)}$ states and $O(k)$ priorities. Using our logarithmic bound of k in the scc-size of games, we show that $\mathcal{A}$ and $\mathcal{A}_k$ are equivalent for $k > \log n$. Then, applying the standard $O(m^d)$ transformation into weak automata [12] (where m is the number of states in the automaton and d the number of its priorities) to $\mathcal{A}_{1+\log n}$ rather than to the original $\mathcal{A}$, we get a translation with a quasi-polynomial blow-up.

For tree automata, the picture is very different. While every alternating parity word automaton can be translated into a weak one, this is not the case with tree automata. In fact there is a strict expressiveness hierarchy of parity tree automata, defined by the automaton's *index*, that is, the number of its priorities [4]. It is known as the Mostowski–Rabin hierarchy in the automata-theoretic literature, and as the alternation hierarchy in the μ-calculus literature. The decidability of whether a given language is expressible in some level of the hierarchy, and specifically by an alternating weak automaton, is open. Here we show that even when a language is recognised by an alternating weak automaton, this automaton may be exponentially larger than an equivalent parity, or even a co-Büchi automaton.

Analogously to the parity hierarchy, there is a hierarchy of weak automata, defined by the number of alternations between accepting and rejecting states. Like the Mostowski–Rabin hierarchy, it also collapses in the word setting [16, 9] and is infinite in the tree setting [19, 21]. So far, its strictness has only been shown for ranked (directed/ordered) trees.

In ranked trees, each child of a node is distinguished by its unique direction – `left` and `right` for binary trees. In unranked (undirected/unordered) trees, which are more common when talking of Kripke structures, and modal or temporal logics, this is not the case. An unranked tree automaton (also known as a symmetric tree automaton) can only require that there exists a child ($\Diamond$) with some property and that all children ($\Box$) have some property. It thus cannot recognise properties such as "there are two distinct children that satisfy p".

We extend the proof of the strictness of the weak hierarchy to alternating automata that run on unranked trees. Our proof combines the technique used for ranked trees in [21] and a weak version of the languages known to be strict for the parity hierarchy [7, 3]. We show

that the language of unranked trees that represent co-Büchi games in which Eve wins with a strategy that sees at most n alternations between accepting and rejecting positions is only recognised by alternating weak automata of level at least n in the hierarchy.

We use the strictness of the weak hierarchy as an intermediate step in showing that there is at least an exponential size blow-up in translating a universal co-Büchi tree automaton into an alternating weak one. We define a language of unranked trees recognised by an automaton of the former type of size in $O(n)$, and show that it is strict for the $(n+1)2^n$ level of the weak hierarchy. This language combines the aforementioned language of trees representing games in which Eve wins in a restricted way, and explicit counting with n binary bits.

The lower bound we present on unranked tree automata also holds for ranked tree automata: the tree languages we work with can easily be translated into languages of ranked binary trees, and the automata that we construct, or argue that do not exist, can be adapted accordingly. Indeed, the trees we build in the proofs are already binary; and the automata that operate on them can be turned into automata on ranked binary trees by transforming $\Box q$ in transition conditions into $(\texttt{left}, q) \wedge (\texttt{right}, q)$ and $\Diamond q$ into $(\texttt{left}, q) \vee (\texttt{right}, q)$.

Related Work. Over words, the best known upper bound for the size blow-up involved in translating alternating parity to alternating weak automata is exponential [12]. The known lower bound, $\Omega(n \log n)$, is the lower bound for translating alternating Büchi into weak automata [13]. It is directly linked to the $2^{\Omega(n \log n)}$ lower bound in determinizing nondeterministic Büchi automata [18]. How to use the power of the parity condition and the limitations of alternation (as opposed to concurrency) to get a better lower bound is open.

Over trees, little is known about the decidability of the weak definability of languages recognised by alternating parity automata. It is known that the intersection of Büchi and co-Büchi definable languages is weakly definable [11]. Weak definability is decidable for tree languages recognised by alternating Büchi automata, as was first shown for ranked trees by Colcombet et al [6] via reduction to cost automata. Skrzypczak and Walukiewicz [22] provide a topological characterisation and exponential decision procedure, which was later extended to unranked trees [15]. It seems that a singly-exponential translation of universal co-Büchi to alternating weak tree automata can be extracted from these procedures to match our lower bound; however it is a non trivial procedure that is yet to be verified.

Due to space constraints, some of the proofs are omitted and appear in the appendix.

2 Preliminaries

Alphabets, words, and trees. An alphabet is an arbitrary finite and nonempty set, usually denoted by Σ. We also use two specific alphabets: $\Sigma_G = \{E_0, A_0, E_1, A_1\}$, to which we refer as the *game alphabet* and $\Sigma_B = \{\texttt{0}, \texttt{1}, \$\}$, to which we refer as the *binary-counting alphabet*.

A *word* over Σ is a (possibly infinite) sequence $w = w_0 \cdot w_1 \cdots$ of letters in Σ. We write $\mathsf{suffixes}(w)$ for the set of suffixes of w (which includes w itself). We consider a *tree* to be an unranked infinite rooted tree in the graph-theoretic sense. A Σ-*tree* is a tree together with a mapping of each of its nodes to a letter in Σ.

For a word or tree language L over an alphabet Σ, we denote by $\overline{L}$ its complement, namely the set of words over Σ or Σ-labeled trees that are not in L.

For natural numbers i and j, we write $[i..j]$ for the set of natural numbers between them, including i and j. For a set Q, we denote by $\mathsf{B}^+(Q)$ the set of positive boolean formulas over the atomic propositions $Q \cup \{\texttt{true}, \texttt{false}\}$.

Automata. Several definitions of alternating tree automata exist. The differences relate to how flexible the transition condition is with respect to ϵ-transitions and the combination of path quantifiers $(\Diamond, \Box)$ and boolean connectives $(\vee, \wedge)$. Usually, all of these definitions give the same expressiveness ([24, Proposition 1] and [10, Remark 9.4]), except for the case of very restricted automata, in which they do not [2]. In our definition of alternating automata, there are no epsilon transitions and path quantifiers are applied directly on states.

An *alternating tree automaton* is a tuple $\langle \Sigma, Q, \iota, \delta, \Omega \rangle$ where Σ is a finite alphabet, Q is a finite set of states, $\iota \in Q$ is the initial state, $\delta : Q \times \Sigma \to \mathsf{B}^+(\{\Diamond, \Box\} \times Q)$ is the transition function, and Ω is the acceptance condition, on which we further elaborate below.

Intuitively, given a state $q \in Q$ and a letter $\sigma \in \Sigma$, the transition function returns a positive boolean formula that defines which states the automaton should transition to, and whether to consider the next state at one non-deterministically chosen child $(\Diamond)$, or at all of the children $(\Box)$. Positive boolean formulas over $\{\Diamond, \Box\} \times Q$ are called *transition conditions*.

Formally, a *run* of an automaton with states Q over a Σ-tree t is a $(Q \times \Sigma)$-tree r that assigns states to nodes of t along the transition function of $\mathcal{A}$. That is, there is a binary relation ρ that relates nodes of t and nodes of r and satisfies the following constraints.

- For every pair $(n, n') \in \rho$, n is a node of t, n' is a node of r, and if n is labeled σ then n' is labeled $(\cdot, \sigma)$. (The $\cdot$ stands for an arbitrary value.) For every node n' of r, there is exactly one node n of t, such that $(n, n') \in \rho$.
- The roots of t and of r appear in exactly one pair, together, and r's root is labeled $(\iota, \cdot)$.
- For a node n of t with parent p, and a node n' of r with parent p', if $(n, n') \in \rho$ then $(p, p') \in \rho$.
- Consider a node n of t, and a node n' of r labeled (q, σ), such that $(n, n') \in \rho$. Let $\Phi = \delta(q, \sigma)$ be the transition condition of the state q over the letter σ. Then Φ should be satisfied by ρ as inductively defined below.
  - If $\Phi = \Diamond h$ then there exists a child c of n and a child c' of n', such that $(c, c') \in \rho$ and c' is labeled $(h, \cdot)$.
  - If $\Phi = \Box h$ then for every child c of n, there is a child c' of n', such that $(c, c') \in \rho$ and c' is labeled $(h, \cdot)$.
  - If $\Phi = b_1 \vee b_2$ (resp. $\Phi = b_1 \wedge b_2$), for transition conditions b_1 and b_2, then b_1 or b_2 (resp. b_1 and b_2) should be satisfied.

A run is *accepting* if each of its paths satisfies the acceptance condition Ω or ends with `true`. There exist various acceptance conditions in the literature; We use the following.

- *Büchi* (resp. *co-Büchi*), where $\Omega \subseteq Q$ is the set of *accepting states*, and a path is accepting if some state (resp. all states) that it visits infinitely often are in Ω.
- A Büchi (and a co-Büchi) automaton is *weak* if every strongly connected component in the transition graph consists of either only accepting states or only rejecting states.
- *Parity*, where $\Omega : Q \to I$ is a *priority function* that assigns to each state a priority from a set $I = [0..i]$ or $I = [1..i]$, for some $i \in \mathbb{N}$. A path is accepting if the maximal priority seen infinitely often in it is even.

Note that the weak condition is a special case of both the Büchi and co-Büchi conditions, which are dual and are both special cases of the parity condition.

An automaton $\mathcal{A}$ *accepts* a tree if it has an accepting run on it; the language that it *recognises*, denoted by $L(\mathcal{A})$, is the set of trees that it accepts. Two automata that recognise the same language are *equivalent*.

The *size* of an automaton is the maximum of the alphabet length, the number of states, the number of subformulas in the transition function, and the acceptance condition's *index*, which is 1 for Büchi and co-Büchi, and $|I|$ for parity. Observe that in alternating automata,

the difference between the size of an automaton and the number of states in it can stem from a transition function that has exponentially many subformulas. (In stronger acceptance conditions, not considered here, the index might also be exponential in the number of states.)

Nondeterminism in tree automata also varies in the literature. In general, it only concerns the boolean connectives of the transition condition and not the path quantifiers (or directions, in ranked trees). We consider an alternating automaton to be *nondeterministic* (resp. *universal*) if its transition conditions only use the $\vee$ (resp. $\wedge$) connective, in addition to the path quantifiers $\Diamond$ and $\Box$.

Word automata are defined exactly as tree automata, without the path quantifiers $\Diamond$ and $\Box$. Accordingly, a run of a word automaton is a sequence of states.

The *class* of an automaton characterises its transition mode (deterministic, nondeterministic, or alternating), its acceptance condition, and whether it runs on words or trees. We often abbreviate automata classes by acronyms in $\{\text{D, N, A}\} \times \{\text{W, B, C, P}\} \times \{\text{W, T}\}$. The first letter stands for the transition mode; the second for the acceptance-condition (weak, Büchi, co-Büchi, and parity); and the third indicates whether the automaton runs on Words or on Trees. For example, AWW stands for an alternating weak automaton on words.

It is known that AWWs recognise all ω-regular word languages [16], while AWTs do *not*: they have the same expressiveness as alternation-free μ-calculus (AFMC) [20].

Games. A *parity game* is an infinite-duration path-forming game, played between Eve and her opponent Adam on a game graph $G = \langle V, V_e, V_a, E, \Omega\rangle$ called the *arena*. The positions V of the arena are partitioned into those belonging to Eve, V_e, and those belonging to Adam, V_a. We assume that every position has at least one successor. The *priority assignment* $\Omega \to I$ maps every position to a *priority* in I, a finite prefix of the non-negative integers. Starting at some position of G, a *play* proceeds with the owner of the current position choosing the next position among its successors in the directed edge relation $E \subseteq V \times V$. The players collaboratively form a play, consisting of an infinite path along the edges of the game graph. Eve wins if the highest priority visited infinitely often is even, and Adam wins if it is odd.

A *co-Büchi game* is a parity game, in which the set of priorities is $I = \{0, 1\}$. (The positions with priority 0 are *accepting* and those with priority 1 are *rejecting*.)

We shall view a Σ_G-tree t also as a co-Büchi game, where nodes labelled E_i and A_i, for $i \in \{0, 1\}$, are interpreted as Eve's and Adam's positions respectively, and have priority i. If not stated differently, we assume that the game starts at the root of t.

A (positional) *strategy* σ for a player $P \in \{\text{Adam,Eve}\}$ maps every position v belonging to P to one of its successors $\sigma(v)$. A play $\pi = v_0 v_1 \dots$ is said to agree with σ when for all i, if v_i belongs to P, then v_{i+1} is $\sigma(v_i)$. A strategy σ for player P is said to be winning for P from a region $W \subseteq V$ if all plays starting within W that agree with σ are winning for P.

▶ **Proposition 1** (Positional determinacy [7]). *Parity games are positionally determined: at each position, either Adam or Eve has a positional winning strategy.*

▶ **Definition 2** (Model-checking game). Given a word $w \in \Sigma^\omega$ and an APW $\mathcal{A} = \langle \Sigma, Q, \iota, \delta, \Omega\rangle$, the model-checking game $\mathcal{G}(w, \mathcal{A})$ is the following parity game:

- Positions are $\mathsf{B}^+(Q) \times \mathsf{suffixes}(w)$.
- For $a \in \Sigma$, $u \in \Sigma^\omega$ and transition conditions b and b', there is an edge from:
  - $(b \wedge b', u)$ to (b, u) and (b', u)
  - (q, au) to $(\delta(q, a), u)$
  - $(\texttt{true}, au)$ to $(\texttt{true}, u)$
  - $(b \vee b', u)$ to (b, u) and (b', u)
  - $(\texttt{false}, au)$ to $(\texttt{false}, u)$.

- Positions $(b \wedge b', u)$ belong to Adam, other positions belong to Eve.
- A position (b, u) is of priority $\Omega(b)$ if b is a state, 1 if $b = \texttt{false}$, and 0 otherwise.

Note that for ultimately periodic words w, the game $\mathcal{G}(w, A)$ has finitely many positions.

▶ **Proposition 3.** *An APW $\mathcal{A}$ with initial state ι accepts a word w if and only if Eve has a winning strategy in the model-checking game $\mathcal{G}(w, \mathcal{A})$ from (ι, w).*

3 The Weak Hierarchy

A weak automaton can be categorised by the maximal number of alternations[3] between accepting and rejecting states in it [19]. This is a strict hierarchy over ranked trees [19, 21], whereas over words it collapses [16, 9]. We clarify the level to which it collapses over words, and extend the strictness result to the setting of unranked trees. We will use this hierarchy to show the lower bound for translating alternating parity to alternating weak tree automata.

Formally, to define its level, we consider a weak automaton as a parity automaton in which whenever a state q is reachable from a state q', $\Omega(q) \leq \Omega(q')$, *i.e.* the parities seen on any path of the automaton are non-increasing. In such parity automata, there are no cycles with both an even and an odd priority, and therefore all even priorities can be replaced with 0 and odd priorities with 1. This definition therefore coincides with the definition of a weak automaton as a special case of a Büchi or co-Büchi automaton, as given in Section 2, by interpreting states of even priority as accepting and odd priority as rejecting. Then, for every $n \in \mathbb{N}$, a $(0, n)$-*AWT* (resp. $(1, n+1)$-*AWT*) is a weak automaton with priorities (ranks) within $[0..n]$ (resp. $[1..n+1]$). Notice that every AWT can also have transitions to `true` and `false`, which "do not count" in the ranking

The classes of $(0, n)$- and $(1, n+1)$-AWTs form the *weak hierarchy*. Notice that for every $n \in \mathbb{N}$, an automaton is a $(0, n)$-AWT iff its dual is a $(1, n+1)$-AWT, and the class of $(1, n+1)$-AWTs is contained in the class of $(0, n+1)$-AWTs.

The weak hierarchy over words is defined analogously for AWWs rather than AWTs.

3.1 Word Automata: Collapse of the Hierarchy

AWWs recognise all ω-regular word languages: there is a translation of deterministic Muller word automata into AWWs $\mathcal{A}$ [16]. A close look at the construction reveals that $\mathcal{A}$ is in the $(1, 3)$ class. We now show that this is the first class that recognises all ω-regular languages.

▶ **Theorem 4.** *Every ω-regular word language is recognised by some $(0, 2)$-AWW, and there are ω-regular word languages not recognised by any $(0, 1)$-AWW.*

One direction follows from the translation in [16]; for the other direction we show that the Büchi condition, i.e. the language of words with infinitely many repetitions of a fixed letter, is not recognised by any $(0, 1)$-AWW.

3.2 Tree Automata: Strict Hierarchy for Ranked and Unranked Trees

We extend the result on the strictness of the weak hierarchy to the setting of unranked trees. Our proof combines the technique used for ranked trees in [21] and a weak version of the cannonical example of a family of languages known to exhaust the parity hierarchy [7, 3].

[3] This sort of "alternation" has nothing to do with the "alternation" of alternating automata, which refers to having both nondeterminism and universality in the transition condition.

For $n \in \mathbb{N}$, let Win_n be the language of trees over the game alphabet Σ_G in which, when viewed as co-Büchi games, Eve wins with up to n alternations between 0 and 1 nodes. That is, Eve has a winning strategy σ, such that every path of the tree that agrees with σ has up to n transitions from a node labeled E_x/A_x to a node labeled E_y/A_y, for $x \neq y \in \{0, 1\}$.

We start with the positive direction, showing that Win_n is recognised by a $(0, n)$-AWT. The $(0, n)$-AWT is simply the standard automaton to recognise a winning strategy for Eve, duplicated to count alternations.

▶ **Lemma 5.** *For every $n \in \mathbb{N}$, Win_n is recognised by a $(0, n)$-AWT.*

We continue with the negative direction, showing that the complement of Win_n is not recognised by a $(0, n)$-AWT.

▶ **Lemma 6.** *For every $n \in \mathbb{N}$, $\overline{\mathsf{Win}_n}$ is not recognised by a $(0, n)$-AWT.*

The proof consists of an induction on n. For the inductive case we assume, towards a contradiction, that a $(0, n + 1)$-AWT $\mathcal{A}$ recognises $\overline{\mathsf{Win}_{n+1}}$. The key to this proof is that when $n + 1$ is odd, there must be some depth after which an accepting run does not visit states of priority n. We can then use states of lower priority to build a $(0, n)$-AWT that recognises $\overline{\mathsf{Win}_n}$, reaching a contradiction. For odd n, we look at the dual $(1, n + 2)$-AWT in which $n + 2$ is odd and follow the analogous reasoning.

▶ **Theorem 7.** *The weak hierarchy is strict for alternating weak tree automata over unranked trees. That is, for every integer n, there is a language of unranked trees that is recognised by a $(0, n + 1)$-AWT and not by any $(0, n)$-AWT.*

Proof. The family of languages $\overline{\mathsf{Win}_n}$ fits the bill. By Lemma 5, for every $n \in \mathbb{N}$, Win_n is recognised by a $(0, n)$-AWT, implying that $\overline{\mathsf{Win}_n}$ is recognised by a $(1, n + 1)$-AWT, which is in particular a $(0, n + 1)$-AWT. By Lemma 6, $\overline{\mathsf{Win}_n}$ is not recognised by any $(0, n)$-AWT. ◀

4 From Alternating Parity to Alternating Weak Word Automata

We present a quasi-polynomial transformation from APW to AWW, based on the idea of register games [14]. These were developed as an automata-theoretic method for solving parity games in quasi-polynomial time; here we show that this approach is more general and can be used to turn APW into AWW with a quasi-polynomial state and size blow-up.

Register games are a parameterised variant of parity games, also played on a parity game arena. We define for any APW $\mathcal{A}$ an APW $\mathcal{A}_k$ that accepts an ultimately periodic word w if and only if Eve wins the k-register game on the arena of the model-checking game $\mathcal{G}(w, \mathcal{A})$. When $\mathcal{A}$ has n states, $\mathcal{A}_k$ has $O(kn^{k+1})$ states and $2k + 1$ priorities. We then show that $\mathcal{A}$ and $\mathcal{A}_k$ are equivalent for $k = 1 + \log n$. Then, applying the standard $O(m^d)$ transformation into weak automata [12], where m is the number of states in the automaton and d the number of priorities in it, however to $\mathcal{A}_{1+\log n}$ rather than to $\mathcal{A}$, we get a translation with a quasi-polynomial blow-up.

A similar line of reasoning does not work on trees because the register-index of the model-checking game between a tree and an APT depends on both the tree and the automaton.

▶ **Definition 8** (Register game [14])**.** For a strictly positive integer k, a *k-register game* consists of a normal parity game, augmented with k registers. Each register records the highest priority that has occurred in the parity game since it was last reset. The registers are ranked according to how long it has been since their last reset, with a newly reset register having rank 1. Eve is given control of the registers: Before every move in the parity game,

Eve can choose to reset a register of any rank r. If the register contains the priority p, this produces output $2r$ if p is even and $2r+1$ otherwise. As long as Eve resets registers infinitely often, this produces an infinite sequence in $[1..2k+1]^\omega$. To win, Eve has to produce an infinite sequence of outputs such that the maximal value output infinitely often is even. A play in a register game can begin at any position and register configuration (even having registers with an integer bigger than the maximal priority of the parity game); Note that the initial register configuration does not affect the number of registers needed. Unless stated otherwise, we assume all the registers to be initialised to 0.

We call a k-tuple $\bar{x}$ of priorities a *register configuration*, where x_i is the content of the register of rank i. The *top register* is the register of rank k.

▶ **Definition 9** (Register-index [14])**.** The *register-index* of a parity game G from a position v is the least k such that Eve wins either both the k-register game and the parity game on G from v or both the k-register game and the parity game on the dual of G from v (i.e. priorities are shifted by 1 and node ownership is inverted). The register-index always exists [14]. The register-index of a region W winning for Eve is the least k such that Eve wins the k-register game from *any* position in W. The register-index of G is the maximal register-index over all its positions.

We now define for every APW $\mathcal{A}$ its parameterised version $\mathcal{A}_k$, which is an APW that will be shown to accept a word w if and only if Eve wins the k-register game on $\mathcal{G}(w,\mathcal{A})$ starting from (ι, w). The idea is to emulate the k-register game by keeping track of register configurations with a tuple $\bar{x} \in I^k$ that is updated according to which priorities are seen and Eve's resetting choices, which are represented as nondeterministic choices in $\mathcal{A}_k$. The outputs from resets are captured by the priorities of the states of $\mathcal{A}_k$. Here we note a slight subtlety: In the k-register game on $\mathcal{G}(w,\mathcal{A})$, Eve can reset not only at positions (q,u) where q is a state of $\mathcal{A}$, but also at positions (b,u) where b is a boolean formula. In $\mathcal{A}_k$ we aggregate the outputs from all resets between two states (by taking the largest among them) into the priority of the next state – this is the third element $p \in [1..2k+1]$ of the states of $\mathcal{A}_k$. When Eve does not reset, we use the priority 1 so that if Eve does not reset infinitely often, she loses the game, as required.

▶ **Definition 10.** Given an APW $\mathcal{A} = \langle \Sigma, Q, \iota, \delta, \Omega\rangle$ with $\Omega : Q \to I$ and a strictly positive integer k, we define an APW $\mathcal{A}_k = \langle \Sigma, Q', \iota', \delta', \Omega'\rangle$ as follows:

- $Q' = Q \times I^k \times [1..2k+1]$
- $\iota' = (\iota, (0,..,0), 1)$
- Ω': For every $q \in Q, \bar{x} \in I^k$, and $p \in [1..2k+1]$, we have $\Omega'(q,\bar{x},p) = p$.
- δ': For every $q \in Q, \bar{y} \in I^k, p \in [1..2k+1]$, and $a \in \Sigma$, we have $\delta'((q,\bar{y},p),a) = \mathsf{move}(\delta(q,a),\bar{y},1) \vee \mathsf{reset}(\delta(q,a),\bar{y},1)$, where for every $q' \in Q, \bar{x} \in I^k, p' \in [1..2k+1]$, and transition conditions b and b':
  - $\mathsf{move}(q',\bar{x},p') = (q',\bar{x}',p')$, with $x'_i = \max(x_i, \Omega(q'))$.
  - $\mathsf{move}(b \wedge b',\bar{x},p') = (\mathsf{move}(b,\bar{x},p') \vee \mathsf{reset}(b,\bar{x},p')) \wedge (\mathsf{move}(b',\bar{x},p') \vee \mathsf{reset}(b',\bar{x},p'))$
  - $\mathsf{move}(b \vee b',\bar{x},p') = (\mathsf{move}(b,\bar{x},p') \vee \mathsf{reset}(b,\bar{x},p')) \vee (\mathsf{move}(b',\bar{x},p') \vee \mathsf{reset}(b',\bar{x},p'))$
  - $\mathsf{reset}(b,\bar{x},p') = \bigvee_{i\in[1..k]} \mathsf{reset}_i(b,\bar{x},p')$, where $\mathsf{reset}_i(b,\bar{x},p') = \mathsf{move}(b,\bar{x}',p'')$ with
    - $x'_j = x_j$ for $j > i$; $x'_j = x_{j-1}$ for $j \le i, j > 1$; $x'_1 = 0$
    - p'' is $\max(2i,p')$ if x_i is even; and $\max(2i+1,p')$ otherwise.

▶ **Lemma 11.** *Given an APW $\mathcal{A}$, the APW $\mathcal{A}_k$ accepts a word w if and only if Eve wins the k-register game on $\mathcal{G}(w,\mathcal{A})$ from (ι, w).*

The register-index of a parity game G is logarithmically bounded in the number of positions in G [14]. However, the number of positions of $\mathcal{G}(w, \mathcal{A})$ depend on both w and $\mathcal{A}$. We introduce a new measure of game size that we use to logarithmically bound the register-index, but which in $\mathcal{G}(w, \mathcal{A})$ only depends on $\mathcal{A}$.

▶ **Definition 12.** For a game G, let S_G be a maximal set of distinct (not necessarily maximal) strongly connected components in G. Call the size of S_G the *scc-size* of G.

The proof that the register-index of a game is logarithmic in its scc-size is similar to the proof that it is logarithmic in the number of positions of the game [14], while strengthening it by showing that: i) every game of scc-size 1 has register-index 1, and ii) as the register-index increases, the scc-size, rather than just the number of positions, has to double.

We use slightly stronger strategies for Eve: instead of just winning, we require her not to reset the top register when it contains an odd priority. This allows us to compose strategies.

▶ **Definition 13** (Defensive register-index [14])**.** A winning strategy for Eve in a register game on G is *defensive* if, whenever the strategy is played from a position in the winning region of G and a register-configuration in which the top register contains an even priority no smaller than the largest priority in G, the play never outputs $2k + 1$.

The *defensive register-index* of a winning region for Eve of a parity game is the lowest integer k such that Eve has a defensive winning strategy. Observe that the defensive register-index is trivially at most one larger than the register-index.

The proof that the defensive register-index is logarithmic in the scc-size is different for arbitrary scc-size and scc-size 1; we consider the two cases separately.

▶ **Lemma 14.** *A parity game with scc-size* 1 *has defensive register-index* 1*.*

Proof sketch. We prove by induction on the number of positions in an arena G that Eve has a defensive strategy in the 1-register game on G, such that if the initial value of the register is 0 then a play that agrees with the strategy outputs only 2's. The base case is trivial.

In the induction step, we consider the maximal strongly connected components consisting of vertices with priority smaller than the maximal priority p. Observe that there is up to one such component, denoted by G_s. If it does not exist, Eve's strategy is to reset whenever p is seen. If G_s does exist, Eve's strategy is to reset the register whenever entering G_s, and within G_s to follow the strategy that is guaranteed by the induction assumption. The correctness builds on the fact that in an arena with scc-size 1, all cycles intersect. ◀

▶ **Lemma 15.** *The register-index k of a parity game of scc-size z is at most $1 + \log z$.*

Proof. From the definition of register-index, it suffices to consider the single-player parity games G induced by any winning strategy for Eve in her winning region. Observe that in the register games on G, Eve's strategy consists of just choosing when to reset registers.

We proceed by induction on the number of positions n in G. The base case, $n = 1$, is trivial. For the inductive step, let G' be the game induced by positions of G of priority up to $p - 2$, where p is the maximal even priority that appears in some cycle of G. Then, let $G_1, \ldots, G_j$ be the maximal strongly connected subgames of G'. Let $k_1, \ldots, k_j$ be their respective defensive register-indices, and k_m the maximal among these. If there are no such subgames, then all cycles in G contain p in which case Eve wins defensively the 1-register game on G by resetting whenever p is seen.

Case of a unique i for which $k_i = k_m$, and $k_m > 1$: We show that the register-index of G is no more than k_m. Since the scc-size of G is at least that of G_i, and using the induction hypothesis, this suffices.

Eve's strategy in the k_m-register game on G is as follows: she first resets any odd priorities in the registers (this may take several turns); then, after this clean-up phase, she resets k_m whenever p occurs; within a subgame G_j she uses the bottom ranking k_j registers to simulate her defensive winning strategy in the k_j-register game on G_j.

Assume that this strategy is played from a register-configuration where the top register contains an even priority no smaller than p. First observe that since $k_m > 1$, after seeing p and resetting k_m, the top register still contains an even value no smaller than p. Furthermore, since this strategy encounters p between any two entrances into G_i, it always enters G_i with an even value no smaller than p in the top register. Thus, it never outputs $2k_m + 1$. It leaves Adam the choice of losing within a subgame G_j (where Eve follows a winning strategy) or changing subgames infinitely often. In the latter case, p is seen infinitely often, thus producing $2k_m$ as output infinitely often. It is therefore winning and defensive for Eve.

If it is played from a register-configuration in which the top register is not an even priority no smaller than p, then it might output $2k_m + 1$, but the values output infinitely often will still be the same as above, so the strategy is still winning and defensive.

Case of i, j where $i \neq j$ and $k_i = k_j = k_m$: We show that the register-index of G is no more than $k_m + 1$. This suffices, since by the induction hypothesis, each of G_i and G_j has scc-size at least $2^{k_m - 1}$; then G has scc-size at least 2^{k_m}.

Eve's strategy in the $k_m + 1$-register game on G is as follows: reset the register of rank $k_m + 1$ whenever p is seen; in a subgame G_i, use the bottom ranking k_i registers to simulate a winning strategy in the k_i-register game on G. As above, this strategy is winning, and since it only resets the register of rank $k_m + 1$ after seeing p, it is defensive.

Case of $k_m = 1$: If the scc-size of G is 1 then the result follows from Lemma 14. Otherwise, Eve can win the 2-register game on G, using a strategy as above: within a subgame, she uses her defensive 1-register strategy, and resets the top register whenever p is seen. This strategy is winning since a play either remains in a subgame and follows a winning strategy, or sees p infinitely often and therefore outputs 4 infinitely often, but does not output 5 infinitely often. It is therefore winning. If initially the top register contains an even priority no smaller than p, this strategy never outputs 5 and is therefore defensive. ◀

We now show that the scc-size of $\mathcal{G}(w, \mathcal{A})$, for an ultimately periodic word w, is independent of w and logarithmic in $\mathcal{A}$. It basically follows from Lemma 15 and the fact that w is represented by a Kripke structure with a single cycle.

▶ **Lemma 16.** *Given an ultimately periodic word $w = uc^\omega$ and an APW $\mathcal{A}$ with n states, the parity game $\mathcal{G}(w, \mathcal{A})$ has register-index at most $1 + \log n$.*

Then $\mathcal{A}$ is equivalent to its $1 + \log n$ parameterised version; the main result follows by applying the existing transformation – exponential in the number of priorities – to $\mathcal{A}_{1+\log n}$.

▶ **Lemma 17.** *Every APW A is equivalent to its parameterised version A_k, for $k = 1 + \log n$.*

▶ **Theorem 18.** *The size blow-up and state blow-up involved in translating alternating parity word automata to alternating weak word automata is at most quasi-polynomial. In particular, every APW $\mathcal{A}$ of size (resp. number of states) n is equivalent to an AWW of size (resp. number of states) $2^{O((\log n)^3)}$.*

5 From Universal Co-Büchi to Alternating Weak Tree Automata

As opposed to the word setting, alternating weak tree automata do not recognise all ω-regular tree languages. Furthermore, we show below that in cases that a translation from an alternating parity automaton, or even a universal co-Büchi automaton, is possible, there might be an exponential size blow-up.

We provide a family $\{L_n\}_{n\geq 1}$ of tree languages, such that L_n is only recognised by an AWT with at least 2^n states, while recognised by a UCT of size in $O(n)$.

The languages L_n are defined over the game alphabet combined with the binary-counting alphabet. Intuitively, a tree t belongs to L_n if Eve wins the co-Büchi game with respect to the game alphabet, and every path of t provides a correct prefix of an n-bit binary counter, with respect to the binary-counting alphabet, when only considering nodes that come right after an alternation between E_0/A_0 and E_1/A_1 labelling. Notice that the latter requirement guarantees that there are up to $(n+1)2^n$ such alternations in every path of t. (There are up to 2^n numbers in an n-bit counter, and with the separator \$ every number takes $n+1$ bits.)

Formally, L_n is the language of $(\Sigma_G \times \Sigma_B)$-labeled trees, such that a tree t is in L_n iff it satisfies the following properties:

I. Considering only the Σ_G labelling of t and viewing the tree as a co-Büchi game, Eve wins.

II. The Σ_G-labelling of t's root is E_0 or A_0.

III. For a string π over Σ, a node of π is in the derived string π' if it is labeled by E_x/A_x and its predecessor is labeled by E_y/A_y for $x \neq y$. Then, for every path π of t, the derived string π', when considering only its Σ_B labelling, should be a correct prefix of an n-bit binary counter, where each n bits of `0/1` are separated by \$. For example, `000$001$01` is a correct prefix, while `000$100` and `000$`$\ldots$`$111$000` are not.

Intuitively, i) L_n is recognised by a small UCT that combines three succinct components, each checking one of the three requirements in the definition of L_n; ii) L_n is AWT-recognizable, since the third requirement guarantees boundedly many alternations between accepting and rejecting states; and iii) L_n is only recognised by an AWT with at least $(n+1)2^n$ states, since L_n will be shown to be in that level of the weak alternation hierarchy.

▶ **Lemma 19.** *For every $n \in \mathbb{N}$, L_n is recognised by a UCT of size in $O(n)$.*

Proof. A UCT for L_n can be defined as the conjunction of three UCTs, $\mathcal{A}_I$, $\mathcal{A}_{II}$, and $\mathcal{A}_{III}$, each checking the corresponding requirement in the definition of L_n.

$\mathcal{A}_I$. The UCT $\mathcal{A}_I$ can be defined by straightforwardly relating E_0 and E_1 to accepting and rejecting nondeterministic states, respectively, and A_0 and A_1 to accepting and rejecting universal states, respectively. Formally, $\mathcal{A}_I = \langle \Sigma, Q_I, \iota_I, \delta_I, \alpha_I \rangle$, where

- $Q_I = \{q_0, q_1\}$.
- $\iota_I = q_0$.
- δ_I. For every $q \in Q$ and $b \in \Sigma_B$, we define: $\delta_I(q, (E_0, b)) = \Diamond q_0$; $\delta_I(q, (E_1, b)) = \Diamond q_1$; $\delta_I(q, (A_0, b)) = \Box q_0$; and $\delta_I(q, (A_1, b)) = \Box q_1$.
- $\alpha_I = q_0$.

A winning strategy of Eve on a game-tree t suggests how to resolve the nondeterminism in $\mathcal{A}_I$, while an accepting run of $\mathcal{A}_I$ on t can be translated to a winning strategy for Eve.

$\mathcal{A}_{II}$. Trivially checking the tree root's labeling.

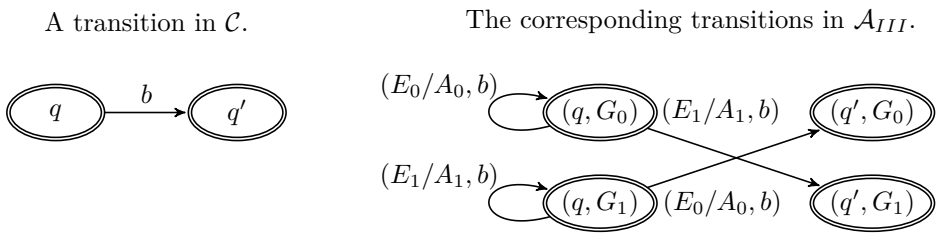

Figure 1 From the universal finite-tree automaton $\mathcal{C}$ to the UCT $\mathcal{A}_{III}$ in the proof of Lemma 19.

$\mathcal{A}_{III}$. The core idea behind the definition of the UCT $\mathcal{A}_{III}$ is that *improper* n-bit binary counting can be recognised by a *nondeterministic* finite-word automaton $\mathcal{A}$ of size in $O(n)$. Furthermore, all states of $\mathcal{A}$ are rejecting, except for a single "forever accepting" state (`true`). See, for example, [1, Figure 2] for a concrete construction of such an automaton.

Accordingly, *proper* n-bit binary counting can be recognised by the dual of $\mathcal{A}$, which is a *universal* finite-word automaton $\mathcal{B}$ of size in $O(n)$. Now, a universal word automaton $\mathcal{A}_w$ for a word language L can be adapted to a universal tree automaton $\mathcal{A}_t$ for the language of trees all of whose paths are in L, by simply adding $\Box$ to every transition in $\mathcal{A}_w$. (Notice that this does not hold for a nondeterministic word automaton.) Hence, we can adapt $\mathcal{B}$ to a universal finite-tree automaton $\mathcal{C}$ that recognises the language of finite trees all of whose paths are a proper prefix of n-bit counter. Notice that all states of $\mathcal{C}$ are accepting, except for a single "forever rejecting" state (`false`).

Now, all that is left for getting $\mathcal{A}_{III}$ is to adapt $\mathcal{C}$ to operate over the extended alphabet and to only consider the binary-counting alphabet when there is an alternation between E_0/A_0 and E_1/A_1 labeling. We do this by having two copies of $\mathcal{C}$, each "remembering" whether the last game-labeling was 0 or 1. This adaptation is illustrated in Figure 1.

Formally, let $\mathcal{C} = \langle \Sigma_B, Q, \iota, \delta, F \rangle$. We define $\mathcal{A}_{III} = \langle \Sigma, Q_{III}, \iota_{III}, \delta_{III}, \alpha_{III} \rangle$, where

- $Q_{III} = q_0 \cup Q \times \{G_0, G_1\}$.
- $\iota_{III} = q_0$.
- δ_{III}.
  - For every $b \in \Sigma_B$, $\delta_{III}(q_0, (E_0/A_0, b)) = \Box(\iota, G_0)$ and $\delta_{III}(q_0, (E_1/A_1, b)) = \Box(\iota, G_1)$.
  - For every $q \in Q$ and $b \in \Sigma_B$, we define:
    - $\delta_{III}((q, G_0), (E_0/A_0, b)) = \Box(q, G_0)$
    - $\delta_{III}((q, G_0), (E_1/A_1, b)) = G_1(\delta(q, b))$, where $G_1(TC)$, for a transition condition TC of $\mathcal{C}$, changes every instance of a state q in TC to (q, G_1). For example, $G_1(\Box q \wedge \Box q') = \Box(q, G_1) \wedge \Box(q', G_1)$.
    - $\delta_{III}((q, G_1), (E_0/A_0, b)) = G_0(\delta(q, b))$, where $G_0(TC)$, for a transition condition TC of $\mathcal{C}$, changes every instance of a state q in TC to (q, G_0).
    - $\delta_{III}((q, G_1), (E_1/A_1, b)) = \Box(q, G_1)$
- $\alpha_{III} = F \times \{G_0, G_1\}$.

Notice that by the special structure of $\mathcal{A}_{III}$, it is not only a UCT and even a $(0, 0)$-UWT. ◀

Since condition III guarantees a bound on the number of alternations between E_0/A_0 and E_1/A_1 labels, the languages $\{L_n\}$ can also be recognised by alternating weak automata.

▶ **Lemma 20.** *For every $n \in \mathbb{N}$, L_n is AWT-recognizable.*

Proof. Analogously to the construction of a UCT for L_n in the proof of Lemma 19, an AWT for L_n can be defined as the conjunction of three AWTs, $\mathcal{B}_I$, $\mathcal{B}_{II}$, and $\mathcal{B}_{III}$, each checking the corresponding requirement in the definition of L_n.

The automata $\mathcal{B}_{II}$ and $\mathcal{B}_{III}$ can be identical to the automata $\mathcal{A}_{II}$ and $\mathcal{A}_{III}$, respectively, from the proof of Lemma 19, as they are already AWTs (and even $(0,0)$-UWTs).

As for $\mathcal{B}_I$, it obviously cannot be identical to $\mathcal{A}_I$ from the proof of Lemma 19, as the latter heavily builds on the co-Büchi acceptance condition. Furthermore, the first requirement in the definition of L_n is not AWT-definable. Yet, due to the third requirement in the definition of L_n, in every tree t that belongs to L_n, all paths have up to $(n+1)2^n$ alternations between E_0/A_0 and E_1/A_1 labelling. Hence, the first requirement in the definition of L_n can be reformulated to "Eve wins t with up to $(n+1)2^n$ alternations between E_0/A_0 and E_1/A_1 nodes", namely to "t belongs to $\mathsf{Win}_{(n+1)2^n}$", without changing L_n.

By Lemma 5, there is an AWT recognizing $\mathsf{Win}_{(n+1)2^n}$, providing the desired AWT $\mathcal{B}_I$. ◀

We establish in two steps the lower bound on the size of an AWT recognizing L_n: i) We prove a claim on the weak-hierarchy levels of a family $\{H_m\}_{m\geq 1}$ of languages that are quite similar to $\{L_n\}$, but lack the explicit counting; and ii) We prove that an AWT for L_n must be in the same level of the weak hierarchy as an AWT for $H_{(n+1)2^n}$.

Formally, H_m is the language of Σ_G-labeled trees, such that a tree t is in H_m iff i) Eve wins t, when viewing t as a co-Büchi game, ii) the root of t is labeled E_0 or A_0 if m is even, and E_1 or A_1 if m is odd, and iii) In every path of t, there are up to m transitions from a node labeled E_x/A_x to a node labeled E_y/A_y, for $x \neq y \in \{0,1\}$.

The proof that H_m is strict for the m^{th} level of the weak hierarchy follows the inductive reasoning of showing that Win_m is in the m^{th} level of the weak hierarchy (Lemma 6), but requires some additional manoeuvres, due to the asymmetry between Adam and Eve in H_m. Specifically, in the induction step, when assuming toward contradiction a $(0,m+1)$-AWT that recognises $\overline{H_{m+1}}$ or a $(1,m+2)$-AWT that recognises H_{m+1}, we not only manipulate subtrees in H_m and $\overline{H_m}$, but also subtrees that are in the conjunction or disjunction of H_m with languages that correspond to the second and third requirements in the definition of H_m.

▶ **Lemma 21.** *For every $m \in \mathbb{N}$, there is no $(0,m)$-AWT recognizing $\overline{H_m}$.*

Proof. We use the following simple AWTs: Let $\mathcal{R}oot0$ (resp. $\mathcal{R}oot1$) be a $(0,0)$-AWT that accepts a tree t iff the root of t is labeled E_0/A_0 (resp. E_1/A_1). For every $m \in \mathbb{N}$, let $\mathcal{A}lt_m$ be a $(0,0)$-AWT that accepts a tree t iff in every path of t, there are up to m transitions from a node labeled E_x/A_x to a node labeled E_y/A_y, for $x \neq y \in \{0,1\}$. Observe that for an even m, a tree $t \in H_m$ iff $t \in L(\mathit{Eve\ wins}) \cap L(\mathcal{R}oot0) \cap L(\mathcal{A}lt_m)$, while t is in $\overline{H_m}$ iff $t \in L(\mathit{Adam\ wins}) \cup L(\mathcal{R}oot1) \cup \overline{L(\mathcal{A}lt_m)}$. We prove the claim by induction on m.

Base case. Recall that there is a $(0,0)$-AWT recognizing $\overline{H_0}$ iff there is a $(1,1)$-AWT recognizing H_0, which is the language of trees in which Eve wins and all paths are labeled A_0/E_0. Let t be the single-path tree labelled A_0 throughout. If a $(1,1)$-AWT $\mathcal{A}$ recognises H_0, it must accept t. Since every loop in $\mathcal{A}$ is rejecting, an accepting run r of $\mathcal{A}$ on t only has finite paths, ending with `true`. Let k be the length of the longest one. Let t' be the tree identical to t up to depth k and labelled A_1 from there on. Then $\mathcal{A}$ accepts t', but $t' \notin H_0$.

Induction step. The induction hypothesis is that $\overline{H_m}$ is not recognised by a $(0,m)$-AWT (and its dual, that H_m is not recognised by a $(1,m+1)$-AWT). There are two cases to consider, an even m and an odd m.

Even m. Assume, towards a contradiction, a $(0, m+1)$-AWT $\mathcal{A}$ that recognises $\overline{H_{m+1}}$. We will build a $(0, m)$-AWT that recognises $\overline{H_m}$, reaching a contradiction.

Consider an arbitrary tree $t' \in L(Adam\ wins) \cap L(\mathcal{R}oot0) \cap L(\mathcal{A}lt_m)$. Let t be the tree consisting of one branch along which all nodes are labelled E_1, and they all have a child labelled E_0 that in turn has a copy of t' as its unique child.

First, we claim that $t \in \overline{H_{m+1}}$: if Eve does not play into a copy of t', the play sees only labels E_1 and is winning for Adam; if Eve plays into a copy of t', then from there she loses.

Now let r be an accepting run of $\mathcal{A}$ on t. As $m+1$ is odd, there is a depth k of t, starting from which r only assigns states of rank at most m. Let $S_{t'}$ be the set of states that r assigns to the root of t' at depth k, and let $\mathcal{A}_{t'}$ be the automaton that is derived from $\mathcal{A}$ by setting $S_{t'}$ to be the initial set, namely having the initial formula $\bigwedge_{q \in S_{t'}} q$ (which can be translated to an initial state). $\mathcal{A}_{t'}$ is a $(0, m)$-AWT, since it lacks the $m+1$-ranked states of $\mathcal{A}$.

Clearly, $\mathcal{A}_{t'}$ accepts t'. Furthermore, $\mathcal{A}_{t'}$ does not accept any tree in H_m: If it accepted some tree $t'' \in H_m$ then $\mathcal{A}$ would also accept the tree $\hat{t}$ derived from t by replacing the occurrence of t' at depth k with t''. However, $\hat{t}$ is not in $\overline{H_{m+1}}$, since Eve wins it (by going to t''), its root is labeled E_0/A_0, and every path of it has up to $m+1$ alternations.

Let $\mathcal{B}$ be the automaton that is the disjunction of all of these $\mathcal{A}_{t'}$ automata (there are finitely many such automata, as each of them corresponds to a subset of $\mathcal{A}$'s states) for $t' \in L(Adam\ wins) \cap L(\mathcal{R}oot0) \cap L(\mathcal{A}lt_m)$, and let $\mathcal{C}$ be the disjunction of $\mathcal{B}$, $\mathcal{R}oot1$, and $\overline{\mathcal{A}lt_m}$. Observe that $\mathcal{C}$ is a $(0, m)$-AWT.

We claim that $\mathcal{C}$ recognises $\overline{H_m}$, leading to the claimed contradiction.

If $t \in \overline{H_m}$, there are two cases: either $t \in L(\mathcal{R}oot1) \cup \overline{L(\mathcal{A}lt_m)}$ or $t \in L(Adam\ wins) \cap L(\mathcal{R}oot0) \cap L(\mathcal{A}lt_m)$. If $t \in L(\mathcal{R}oot1) \cup \overline{L(\mathcal{A}lt_m)}$ then $\mathcal{C}$ clearly accepts t. Moreover, if $t \in L(Adam\ wins) \cap L(\mathcal{R}oot0) \cap L(\mathcal{A}lt_m)$ then $\mathcal{B}$ and therefore $\mathcal{C}$ accepts t. Hence $\overline{H_m} \subseteq L(\mathcal{C})$.

If $t \in H_m$, then $\mathcal{B}$ does not accepts t since no $\mathcal{A}_{t'}$ accepts t. Furthermore, $t \in L(\mathcal{R}oot0) \cap L(\mathcal{A}lt_m)$ so neither $\mathcal{R}oot1$ nor $\overline{\mathcal{A}lt_m}$ accepts t. Hence $\mathcal{C}$ does not accepts t and $L(\mathcal{C}) = \overline{H_m}$.

Odd m. Observe that $\overline{H_{m+1}}$ is recognised by a $(0, m+1)$-AWT if and only if H_{m+1} is recognised by a $(1, m+2)$-AWT. Assume, towards a contradiction, that there is a $(1, m+2)$-AWT $\mathcal{A}$ that recognises H_{m+1}. We will build a $(1, m+1)$-AWT that recognises H_m, reaching a contradiction.

Consider a tree $t' \in H_m$. Let t be the tree consisting of one branch along which all nodes are labelled A_0, and they each have a child labelled A_1 that in turn has a copy of t' as its unique child.

First, we claim that $t \in H_{m+1}$: i) The label of t's root is in $\{E_0, A_0\}$; ii) Since m is odd, the root of t' is labeled E_1/A_1 and thus there are up to $m+1$ alternations in all paths of t; and iii) Eve wins t: if Adam does not play into a copy of t', the play sees only labels A_0 and is winning for Eve; if Adam plays into a copy of t', then from there Eve has a winning strategy.

Now let r be an accepting run of $\mathcal{A}$ on t. Since $m+2$ is odd, there is a depth k starting from which r only assigns states of rank at most $m+1$. Let $S_{t'}$ be the set of states that r assigns to the root of t' at depth k, and let $\mathcal{A}_{t'}$ be the automaton that is derived from $\mathcal{A}$ by setting $S_{t'}$ to be the initial set. Observe that $\mathcal{A}_{t'}$ is a $(1, m+1)$-AWT, since it lacks the $m+2$-ranked states of $\mathcal{A}$.

Obviously, $\mathcal{A}_{t'}$ accepts t'. Furthermore, $\mathcal{A}_{t'}$ does not accept any tree t'' not in H_m, whose root is labeled E_1/A_1: If it accepted such a tree t'' then $\mathcal{A}$ would also accept the tree $\hat{t}$ that is derived from t by replacing the occurrence of t' at depth k with t''. However, $\hat{t}$ is not in

H_{m+1}, since i) if Adam wins t'' then Adam wins $\hat{t}$; and ii) if there exists a path in t'' with more than m alternations then there exists a path in $\hat{t}$ with more than $m+1$ alternations.

Let $\mathcal{B}$ be the automaton that is the disjunction of all of these $\mathcal{A}_{t'}$ automata, and let $\mathcal{C}$ be the automaton that is the conjunction of $\mathcal{B}$ and $\mathcal{R}oot1$. Observe that $\mathcal{C}$ is a $(1, m+1)$-AWT

We claim that $\mathcal{C}$ recognises H_m, leading to the claimed contradiction. If $t \in H_m$, then $\mathcal{B}$ and therefore $\mathcal{C}$ accepts t. If $t \notin H_m$ and the root of t is labeled E_0/A_0 then $\mathcal{R}oot1$ and therefore $\mathcal{C}$ does not accept t. If $t \notin H_m$ and the root of t is labeled E_1/A_1 then no $A_{t'}$ accepts t, and therefore $\mathcal{B}$ and $\mathcal{C}$ do not accept t. Hence $L(\mathcal{C}) = H_m$. ◀

We now show that an AWT $\mathcal{A}$ for L_n cannot be in a lower level of the weak hierarchy than an AWT $\mathcal{A}'$ for $H_{(n+1)2^n}$, and must therefore have at least $(n+1)2^n$ states. The idea is to construct $\mathcal{A}'$ by having $(n+1)2^n$ adapted copies of $\mathcal{A}$, each properly adding the extra letters of the binary-counting alphabet. We get a much bigger automaton than $\mathcal{A}$, yet on the same level of the weak hierarchy, which provides the desired result.

▶ **Lemma 22.** *For every $n \geq 1$, an AWT recognizing L_n must have at least $(n+1)2^n$ states.*

Proof. Let $N = (n+1)2^n$. Consider an AWT $\mathcal{A}$ that recognises L_n, and let m be the minimal natural number such that $\mathcal{A}$ is in the $(0, m)$-AWT class. We will construct from $\mathcal{A}$ a $(0, m)$-AWT $\mathcal{A}'$ that recognises H_N. This will imply the desired result, as by Lemma 21, m must be at least N.

The idea in the construction of $\mathcal{A}'$ is to follow the transitions of $\mathcal{A}$, while properly adding the extra letters of the binary-counting alphabet. In order to add the letters in a way that matches a proper counting, $\mathcal{A}'$ has N copies of $\mathcal{A}$, each corresponding to a number between 0 and $N-1$. The constructed automaton will be much bigger than $\mathcal{A}$, yet on the same level of the weak hierarchy, which provides the desired result.

Formally, let $\mathcal{A} = \langle \Sigma, Q, \iota, \delta, \Omega \rangle$. (For considering the level of $\mathcal{A}$ in the weak hierarchy, we view it as a parity automaton that satisfies the weakness constraint.) For every $i \in [0..N-1]$, let $\mathsf{bit}(i)$ stand for the letter in Σ_B that appears in the ith position of the n-bit counter. For example, for $n = 3$ and $i = 7$, we have $\mathsf{bit}(7) = \$$, since $\$$ is the letter in the 7th position in the counter `000$001$`$\ldots$.

In addition to the proper letter of Σ_B, $\mathcal{A}'$ should also "remember" whether the last alternation was from E_0/A_0 to $E1/A1$ or vice versa. Yet, this can be derived from the counter position, as in even positions of the counter, the last alternation should be to E_0/A_0, and in odd positions, the other way round. We therefore define the function $\mathsf{NextIndex} : \Sigma_G \times [0..N-2] \to [0..N-1]$, by $\mathsf{NextIndex}(g, i) = i+1$ if (i is even and $g \in \{E_1, A_1\}$ or i is odd and $g \in \{E_0, A_0\}$), while in other cases $\mathsf{NextIndex}(g, i) = i$.

We can now define the AWT $\mathcal{A}' = \langle \Sigma_G, Q', \iota', \delta', \Omega' \rangle$, where

- $Q' = Q \times [0..N-1]$.
- $\iota' = (\iota, 0)$.
- δ': For every $q \in Q$, $i \in [0..N-1]$, and $g \in \Sigma_G$, we have $\delta'((q,i), g) = \mathsf{Next}(\delta(q, (g, \mathsf{bit}(i))))$, where $\mathsf{Next}(TC)$, for a transition condition TC of $\mathcal{A}$, changes every instance of a state q in TC to $(q, \mathsf{NextIndex}(g, i))$.
- Ω': For every $q \in Q$ and $i \in [0..N-1]$, $\Omega'(q, i) = \Omega(q)$.

Observe that $\mathcal{A}'$ is in the same level of the weak hierarchy as $\mathcal{A}$, since the priorities and transitions in $\mathcal{A}'$ are the same as in $\mathcal{A}$, except for having a component number, which does not affect the priority of a state.

It is left to show that $\mathcal{A}'$ recognises H_N. Consider a tree $t' \in H_N$. Let t be the Σ-tree that is derived from t' by adding the binary-counting labelling as follows: i) labelling the

root of t with 0; ii) keeping the binary-counting labelling of the parent node if there was no alternation between E_0/A_0 and E_1/A_1 labelling; and iii) labelling the node with the next letter of a proper binary-counter when there is an alternation between E_0/A_0 and E_1/A_1 labelling. Since t' belongs to H_N, t belongs to L_n and there is some accepting run-tree r of $\mathcal{A}$ on t. Recall that r has labels in $Q \times (\Sigma_G \times \Sigma_B)$. Let r' be the $(Q \times \Sigma_B) \times \Sigma_G$-tree that is derived from r by changing the labelling of every node from $(q, (\sigma_G, \sigma_B))$ to $((q, \sigma_B), \sigma_G)$. Observe that r' is an accepting run of $\mathcal{A}'$ on t'.

As for the other direction, consider a tree t' accepted by $\mathcal{A}'$ via some run-tree r'. Recall that r' has labels in $(Q \times \Sigma_B) \times \Sigma_G$. Let r be the $Q \times (\Sigma_G \times \Sigma_B)$-tree that is derived from r' by changing the labelling of every node from $((q, \sigma_B), \sigma_G)$ to $(q, (\sigma_G, \sigma_B))$. Observe that r is an accepting run of $\mathcal{A}$ on some Σ-tree t, such that t is identical to t' except for having additional Σ_B labelling to each node. Hence, $t \in L_n$ and $t' \in H_N$. ◀

Gathering the lemmas above, we get the blow-up involved in the translation.

▶ **Theorem 23.** *The size blow-up and state blow-up involved in translating UCTs to AWTs, when possible, is in* $2^{\Omega(n)}$.

6 Conclusions

Up until the recent quasi-polynomial parity game algorithms, the best known blow-up of turning APW into AWW roughly matched the complexity of known algorithms for solving parity games. Here we have again closed this gap by establishing a quasi-polynomial APW to AWW transformation. This immediately improves the worst-case complexity of parity game solving via reduction to the AWW emptiness problem [23] to quasi-polynomial. While a more effecient APW to AWW transformation will always yield a more efficient parity game solving algorithm, the extent to which the converse holds is an open question.

In stark contrast, our lower bound for transformations to weak tree automata shows that on trees the simplicity of the weak acceptance condition comes at a much higher cost.

References

1 U. Boker and O. Kupferman. Translating to Co-Büchi Made Tight, Unified, and Useful. *ACM Trans. Comput. Log.*, 13(4):29:1–29:26, 2012.

2 U. Boker and Y. Shaulian. Automaton-Based Criteria for Membership in CTL. In *Proceedings of LICS*, pages 155–164, 2018.

3 J. Bradfield. Simplifying the modal mu-calculus alternation hierarchy. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 39–49. Springer, 1998.

4 J.C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, 1998.

5 C. S. Calude, S. Jain, B. Khoussainov, L. Bakhadyr, W. Li, and F. Stephan. Deciding Parity Games in Quasipolynomial Time. In *Proceedings of STOC*, pages 252–263. ACM, 2017.

6 T. Colcombet, D. Kuperberg, C. Löding, and M. Vanden Boom. Deciding the weak definability of Büchi definable tree languages. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 23. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

7 E.A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *Proceeding of FoCS*, pages 368–377. IEEE, 1991.

8 M. Jurdzinski and R. Lazic. Succinct progress measures for solving parity games. In *Proceedings of (LICS)*, pages 1–9, 2017.

9 R. Kaivola. Axiomatising linear time mu-calculus. In *International Conference on Concurrency Theory*, pages 423–437. Springer, 1995.

10 D. Kirsten. *Alternating Tree Automata and Parity Games*, pages 153–167. Springer Berlin Heidelberg, 2002.

11 O. Kupferman and M. Y. Vardi. $\Pi_2 \bigcap \Sigma_2 \equiv$ AFMC. In *Proceedings of ICALP*, pages 697–713. Springer, 2003.

12 O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proceedings of STOC*, pages 224–233, 1998.

13 O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(2):408–429, 2001.

14 K. Lehtinen. A modal μ perspective on solving parity games in quasipolynomial time. In *Proceedings of LICS*, 2018.

15 K. Lehtinen and S. Quickert. Σ_2^μ is decidable for Π_2^μ. In *Conference on Computability in Europe*, pages 292–303. Springer, 2017.

16 P. A. Lindsay. On Alternating omega-Automata. *J. Comput. Syst. Sci.*, 36(1):16–24, 1988.

17 R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and control*, 9(5):521–530, 1966.

18 M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.

19 A.W. Mostowski. Hierarchies of weak automata and weak monadic formulas. *Theoretical Computer Science*, 83(2):323–335, 1991.

20 D.E. Muller and P.E. Schupp. Simulating Alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

21 D. Niwiński and I. Walukiewicz. A gap property of deterministic tree languages. *Theoretical Computer Science*, 303(1):215–231, 2003.

22 M. Skrzypczak and I. Walukiewicz. Deciding the topological complexity of Büchi languages. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

23 A. Di Stasio, A. Murano, G. Perelli, and M. Y. Vardi. Solving parity games using an automata-based algorithm. In *International Conference on Implementation and Application of Automata*, pages 64–76. Springer, 2016.

24 T. Wilke. CTL$^+$ is exponentially more succinct than CTL. In *Proc. of FoSSaCS*, volume 1738 of *LNCS*, pages 110–121. Springer, 1999.

A Appendix – Omitted Proofs

A.1 Proofs of Section 3

▶ **Theorem 4.** *Every ω-regular word language is recognised by some $(0,2)$-AWW, and there are ω-regular word languages not recognised by any $(0,1)$-AWW.*

Proof. In [16], there is a translation of an arbitrary deterministic Muller automaton to a $(1,3)$-AWW. Hence, given an ω-regular language L, one can represent its dual language $\overline{L}$ by a deterministic Muller automaton $\mathcal{A}$ and translate it to an equivalent $(1,3)$-AWW $\mathcal{B}$. The dual of $\mathcal{B}$ is a $(0,2)$-AWW recognizing L.

As for the negative part, consider the language L over the alphabet $\Sigma = \{a,b\}$, consisting of words with infinitely many a's. Observe that the word $w = ababbabbbab^4ab^5\ldots$ is in L.

Assume towards contradiction a $(0,1)$-AWW $\mathcal{A}$ recognizing L, and having a set of states Q. Then, $\mathcal{A}$ accepts w via some run r. Recall that r is a $Q \times \Sigma$-tree. For every $n \in \mathbb{N}$, let S_n be the set of states of $\mathcal{A}$ that appear in the labeling of nodes of r at the level n.

Since Q is finite and the sequences of subsequent b's in w are unbounded, there is a set $S \subseteq Q$ and two numbers $x < y \in \mathbb{N}$, such that there are only b's between the positions x and y of w, $S = S_x = S_y$, and $S = S_n$ for infinitely many $n \in \mathbb{N}$.

As $\mathcal{A}$ is weak and its top rank is rejecting, no state of S can be from the top rank. (Otherwise there is a path in r all of whose nodes are from the top rank, implying that the path is rejecting.) Hence, all states of S are from the 0 rank. Thus, from every node v in the x level of r, there is a run on the finite word b^{y-x}, such that every state that appears in its labeling is from the 0-level of $\mathcal{A}$, and all its leaves are labeled with states from S or `true`.

Let w' be the infinite word that is the same as w up to position x, and from there on has only b's. Then $\mathcal{A}$ has an accepting run r' on w' that is the same as r up to level y, and then for every $i \in \mathbb{N}$ and node v in a $y + i(y - x)$ level that is labeled with some state q, it makes the next $y - x$ steps from v as the run r from a node in its x level that is labeled with q. Yet, $w' \notin L$, leading to a contradiction. ◀

▶ **Lemma 5.** *For every $n \in \mathbb{N}$, Win_n is recognised by a $(0, n)$-AWT.*

Proof. We first define a $(0, n)$-AWT $\mathcal{W}_n$ and then show that it recognises Win_n.
$\mathcal{W}_n$:

- Alphabet: Σ_G (Namely, $\{E_0, E_1, A_0, A_1\}$.)
- States: $\{q_0, q_1, \ldots, q_n\}$
- Initial state: q_n
- Priorities (ranks): $\Omega(q_j) = j$ for all j
- Transitions:
  $\delta(q_0, E_0) = \Diamond q_0$; $\delta(q_0, A_0) = \Box q_0$; $\delta(q_0, E_1/A_1) =$ `false`
  For every $j \in [1..n]$:

$$\delta(q_j, E_0) = \begin{cases} \Diamond q_j & n \text{ is even} \\ \Diamond q_{j-1} & n \text{ is odd} \end{cases}$$

$$\delta(q_j, E_1) = \begin{cases} \Diamond q_j & n \text{ is odd} \\ \Diamond q_{j-1} & n \text{ is even} \end{cases}$$

$$\delta(q_j, A_0) = \begin{cases} \Box q_j & n \text{ is even} \\ \Box q_{j-1} & n \text{ is odd} \end{cases}$$

$$\delta(q_j, A_1) = \begin{cases} \Box q_j & n \text{ is odd} \\ \Box q_{j-1} & n \text{ is even} \end{cases}$$

Observe that $\mathcal{W}_n$ is indeed a $(0, n)$-AWT.

Next, we show that $\mathcal{W}_n$ recognises Win_n. The transition relation guarantees that in every infinite run of $\mathcal{W}_n$ on any tree t, whenever a node v is labeled E_0/A_0 (resp. $E1/A1$), every successor of v is seen at a state of $\mathcal{W}_n$ of even (resp. odd) rank.

Let σ be a winning strategy for Eve in t, such that any play that agrees with σ only sees up to n alternations. We translate σ into an accepting run r_σ of $\mathcal{W}_n$ on t by resolving the nondeterminism of $\mathcal{W}_n$ at a node $v \in t$ labelled E_i along $\sigma(v)$. That is, the choice of a successor node in a $\Diamond$-transition is done along the choice of Eve in σ. Since no play that agrees with σ sees more than n alternations, r_σ does not reach `false`. Furthermore, since every path of t that agrees with σ sees only finitely many nodes labeled E_1/A_1, the run r_σ is accepting.

Conversely, an accepting run r of $\mathcal{W}_n$ on a tree t translates into a strategy σ for Eve on the game t: first observe that r visits up to one state for every node of t, and up to one successor of every node labelled E_0/E_1; Then, at node v labelled E_0/E_1, the strategy σ chooses the unique successor that r visits. Since r is accepting, every path of r can have only finitely many nodes labeled E_1/A_1, implying that every play that agrees with σ sees E_1/A_1 only finitely often. ◀

▶ **Lemma 6.** *For every $n \in \mathbb{N}$, $\overline{\mathsf{Win}_n}$ is not recognised by a $(0,n)$-AWT.*

Proof. We prove the claim by induction on n.

Base case. Recall that there is a $(0,0)$-AWT recognizing $\overline{\mathsf{Win}_0}$ iff there is a $(1,1)$-AWT recognizing Win_0, which is the language of trees in which Eve wins without ever seeing A_1/E_1. Let t be the single-path tree labelled A_0 throughout. If a $(1,1)$-AWT $\mathcal{A}$ recognises Win_0, it must accept t. Since every loop in $\mathcal{A}$ is rejecting, an accepting run r of $\mathcal{A}$ on t only has finite paths, ending with `true`. Let k be the length of the longest one. Let t' be the tree identical to t up to depth k and labelled A_1 from there on. Then $\mathcal{A}$ accepts t', but $t' \notin \mathsf{Win}_0$.

Induction step. The induction hypothesis is that $\overline{\mathsf{Win}_n}$ is not recognised by a $(0,n)$-AWT (and its dual, that Win_n is not recognised by a $(1,n+1)$-AWT). There are two cases to consider, an even n and an odd n.

Even n. Assume, towards a contradiction, a $(0,n+1)$-AWT $\mathcal{A}$ that recognises $\overline{\mathsf{Win}_{n+1}}$. We will build a $(0,n)$-AWT that recognises $\overline{\mathsf{Win}_n}$, reaching a contradiction.

Consider an arbitrary tree $t' \in \overline{\mathsf{Win}_n}$. Let t be the tree consisting of one branch along which all nodes are labelled E_1, and they all have a child labelled E_0 that in turn has a copy of t' as its unique child.

First, we claim that $t \in \overline{\mathsf{Win}_{n+1}}$: if Eve does not play into a copy of t', the play sees only labels E_1 and is winning for Adam; if Eve plays into a copy of t', then from there she either loses or wins but sees over n alternations within t' and therefore over $n+1$ alternations overall.

Now let r be an accepting run of $\mathcal{A}$ on t. Since $n+1$ is odd, there is a depth k of t, starting from which r only assigns states of rank at most n. Let $S_{t'}$ be the set of states that r assigns to the root of t' at depth k, and let $\mathcal{A}_{t'}$ be the automaton that is derived from $\mathcal{A}$ by setting $S_{t'}$ to be the initial set, namely having the initial formula (which can be translated to an initial state) $\bigwedge_{q \in S_{t'}} q$. Observe that $\mathcal{A}_{t'}$ is a $(0,n)$-AWT, since it lacks the n+1-ranked states of $\mathcal{A}$.

Obviously, $\mathcal{A}_{t'}$ accepts t'. Furthermore, $\mathcal{A}_{t'}$ does not accept any tree out of $\overline{\mathsf{Win}_n}$, namely in Win_n: If it accepted some tree $t'' \in \mathsf{Win}_n$ then $\mathcal{A}$ would also accept the tree $\hat{t}$ that is derived from t by replacing the occurrence of t' at depth k with t''. However, $\hat{t}$ is not in $\overline{\mathsf{Win}_{n+1}}$ – Notice that since the last alternation on any play winning for Eve must be from E_1/A_1 to E_0/A_0, in t'' Eve either wins while seeing only up to $n-1$ alternations or the root of t'' is labelled A_0/E_0. Thus, $\hat{t}$ is not in $\overline{\mathsf{Win}_{n+1}}$, since if Eve plays to t'', she wins and sees either at most $n-1$ alternations within t'' and at most $n+1$ overall, or up to n alternations within t'' but only one more, i.e. $n+1$ alternations overall.

Since $\mathcal{A}$ is finite, the set $\{S_{t'} | t' \in \overline{\mathsf{Win}_n}\}$ is finite and therefore the disjunction $\bigvee_{t' \in \overline{\mathsf{Win}_n}} \mathcal{A}_{t'}$ of $(0,n)$-AWTs is a $(0,n)$-AWT that recognises $\overline{\mathsf{Win}_n}$ – a contradiction.

Odd n. Observe that $\overline{\mathsf{Win}_{n+1}}$ is recognised by a $(0,n+1)$-AWT if and only if Win_{n+1} is recognised by a $(1,n+2)$-AWT. Assume, towards a contradiction, that there is a $(1,n+2)$-AWT $\mathcal{A}$ that recognises Win_{n+1}. We will build a $(1,n+1)$-AWT that recognises Win_n,

reaching a contradiction. The construction is analogous to the above case of an even n, using the dual trees.

Consider a tree $t' \in \mathsf{Win}_n$. Let t be the tree consisting of one branch along which all nodes are labelled A_0, and they each have a child labelled A_1 that in turn has a copy of t' as its unique child.

First, we claim that $t \in \mathsf{Win}_{n+1}$: if Adam does not play into a copy of t', the play sees only labels A_0 and is winning for Eve; if Adam plays into a copy of t', then from there Eve has a winning strategy, such that every play that agrees with it has up to n alternations. Observe that such a play in t' either sees up to $n-1$ alternations, or begins with A_1/E_1, since the last alternation must be from E_1/A_1 to E_0/A_0. Then, in either cases, the overall play sees up to n alternation.

Now let r be an accepting run of $\mathcal{A}$ on t. Since $n+2$ is odd, there is a depth k starting from which r only assigns states of rank at most $n+1$. Let $S_{t'}$ be the set of states that r assigns to the root of t' at depth k, and let $\mathcal{A}_{t'}$ be the automaton that is derived from $\mathcal{A}$ by setting $S_{t'}$ to be the initial set. Observe that $\mathcal{A}_{t'}$ is a $(1, n+1)$-AWT, since it lacks the n+2-ranked states of $\mathcal{A}$.

Obviously, $\mathcal{A}_{t'}$ accepts t'. Furthermore, $\mathcal{A}_{t'}$ does not accept any tree not in Win_n: If it accepted some $t'' \notin \mathsf{Win}_n$ then $\mathcal{A}$ would also accept the tree $\hat{t}$ that is derived from t by replacing the occurrence of t' at depth k with t''. However, $\hat{t}$ is not in Win_{n+1}, since if Adam plays to t'', he either wins or forces to see more than n alternations within t'' and therefore more than $n+1$ alternations overall.

Since $\mathcal{A}$ is finite, the set $\{S_{t'} | t' \in \overline{\mathsf{Win}_n}\}$ is finite and therefore the disjunction $\bigvee_{t' \in \overline{\mathsf{Win}_n}} \mathcal{A}_{t'}$ of $(1, n+1)$-AWTs is a $(1, n+1)$-AWT that recognises Win_n – a contradiction. ◀

A.2 Proofs of Section 4

▶ **Lemma 11.** *Given an APW $\mathcal{A}$, the APW $\mathcal{A}_k$ accepts a word w if and only if Eve wins the k-register game on $\mathcal{G}(w, \mathcal{A})$ from (ι, w).*

Proof. Recall that by Proposition 3, $\mathcal{A}$ accepts a word w if and only if Eve wins the parity game $\mathcal{G}(w, \mathcal{A})$ from (ι, w). The intuition is that $\mathcal{G}(w, \mathcal{A}_k)$ encodes as a parity game the k-register game on $\mathcal{G}(w, \mathcal{A})$ by encoding the register-configuration in the state space, Eve's resetting choices as new disjunctions and the highest output from resets between two states as priorities.

Positions of $\mathcal{G}(w, \mathcal{A}_k)$ are in $\mathsf{B}^+(Q \times I^k \times [1..2k+1]) \times \Sigma^\omega$ while configurations of the k-register game on $\mathcal{G}(w, \mathcal{A})$ consist of a position $\mathsf{B}^+(Q) \times \Sigma^\omega$ and a tuple of register values.

$\mathcal{G}(w, \mathcal{A}_k)$ begins at $((\iota, (0, .., 0), 1), w)$ while the k-register game on $\mathcal{G}(w, \mathcal{A})$ begins at (ι, w) with register configuration $(0, .., 0)$.

Now, observe that at a position (q, au) in $\mathcal{G}(w, \mathcal{A})$ at register configuration $\bar{x}$, Eve has a choice to reset a register, or let the parity game proceed to $(\delta(q, a), u)$. Similarly in $\mathcal{G}(w, \mathcal{A}_k)$, from $((q, \bar{x}, p), au)$ Eve has a choice to either proceed with a move or reset a register.

Whenever Eve decides to proceed with a parity-game move, the decision falls in both games to Adam if $\delta(q, a)$ is a conjunction, and to Eve otherwise. Then Eve again has, in both games, the option between proceeding in the parity game or a reset. Observe that a move (that is, not a reset) only affects the register configuration in the register game on $\mathcal{G}(w, \mathcal{A})$ if it reaches a position (q, j) where q is a state (because intermediate positions have priority 0 in the parity game) and in $\mathcal{G}(w, \mathcal{A}_k)$ a non-reset move only affects $\bar{x}$ if it is $\mathsf{move}(q, \bar{x}, p)$ where q is a state. In both cases, if q is a state, the register configuration is updated to contain the maximum of the old value and the priority of q in A.

If Eve decides to reset a register i in the register game on $\mathcal{G}(w, \mathcal{A})$ at (b, u) with register configuration $\bar{x}$, then this updates the register configuration in the same way as Eve's choice of $\mathsf{reset}_i(b, \bar{x}, p)$ updates $\bar{x}$. Furthermore, the output in $\mathcal{G}(w, \mathcal{A})$ is $2i$ if x_i is even and $2i + 1$ if x_i is odd. Observe that the largest such output between two positions (q, au) and (q', u) is recorded in $\mathcal{G}(w, \mathcal{A}_k)$ as the priority p of $(q', \bar{x}, p)$. This guarantees that the largest priority output infinitely often on a play in the register-game on $\mathcal{G}(w, \mathcal{A})$ matches the highest priority seen infinitely often in the corresponding play in $\mathcal{G}(w, \mathcal{A}_k)$. On the other hand, in a play with finitely many resets, the maximal priority seen in $\mathcal{G}(w, \mathcal{A}_k)$ infinitely often is 1, causing Eve to lose, as required.

Then a winning strategy for Eve in one game translates into a winning strategy for Eve in the other game. ◀

▶ **Lemma 14.** *A parity game with scc-size* 1 *has defensive register-index* 1.

Proof. From the definition of register-index, instead of considering an arbitrary parity game and an arbitrary position of it, it suffices to consider the single-player parity games G induced by any winning strategy for Eve from that position. Observe that in the register games on G, Eve's strategy consists of just resetting the register.

We say that a defensive winning strategy in a 1-register game G is "very defensive" if whenever starting the game with the value 0 in the register, a play that agrees with the strategy does not output 3. (That is, it only outputs 2.)

We prove by induction on the number n of positions in a game G with scc-size 1 that Eve has a very-defensive winning strategy in the 1-register game on G.

The case of $n = 1$ is trivial.

For the inductive case, let G' be the game induced by the positions of G of priority up to $p - 2$, where p is the maximal even priority that appears in some cycle of G. Let G_s be the unique maximal strongly connected component in G'. If such a G_s does not exist, then Eve's strategy in the 1-register game on G is to reset whenever p is seen; since there are no cycles without p, this strategy is winning and very defensive.

If G_s does exist, then by the induction hypothesis, Eve has a very defensive winning strategy σ_s in the 1-register game on G_s. In addition, since G is of scc-size 1, there is no cycle in G that is disjoint to G_s.

Eve's very defensive winning strategy σ in the 1-register game on G is as follows: Reset whenever entering G_s, and follow σ_s within G_s.

Observe that a play that agrees with σ must see p between leaving and entering G_s (else G_s either isn't maximal, or there is a cycle with an odd maximal priority). Hence, when entering G_s, the play outputs 2, the register's content is 0, and the play continues along G_s, which is very defensive. Thus, σ is a very defensive winning strategy: there is no output of value 3, neither when entering G_s nor when remaining in it, and there are infinitely many outputs of value 2, either by the strategy σ_s or by leaving and entering G_s infinitely often. ◀

▶ **Lemma 16.** *Given an ultimately periodic word $w = uc^\omega$ and an APW $\mathcal{A}$ with n states, the parity game $\mathcal{G}(w, \mathcal{A})$ has register-index at most $1 + \log n$.*

Proof. We show that the scc-size of $\mathcal{G}(w, \mathcal{A})$ is at most n. Then, from Lemma 15, its register-index is at most $1 + \log n$.

First note that all strongly connected components of $\mathcal{G}(w, \mathcal{A})$ occur in the subgame $\mathcal{G}(c^\omega, \mathcal{A})$. We consider the graph H that is derived from $\mathcal{G}(c^\omega, \mathcal{A})$ by ignoring the intermediate positions (b, v), where b is a boolean formula between positions of the form (q, v), for a state

q. That is, let H be the graph consisting of just the vertices (q, v) of $\mathcal{G}(c^\omega, \mathcal{A})$, where q is a state. The edges of H connect positions (q, v) and (q', v') if (q', v') is reachable from (q, v) in $\mathcal{G}(c^\omega, \mathcal{A})$ directly, that is, with a path which does not visit yet another position (q'', v'') where q'' is a state.

$\mathcal{G}(c^\omega, A)$ has ssc-size no larger than the graph H: a set of distinct strongly connected components in $\mathcal{G}(c^\omega, A)$ induces a set of strongly connected components in H. If m is the size of the cycle c, then each strongly connected component of H must be of size at least m. H is of size at most mn, so the ssc-size of H, and therefore of $\mathcal{G}(w, \mathcal{A})$, is at most n. ◀

▶ **Lemma 17.** *Every APW A is equivalent to its parameterised version A_k, for $k = 1 + \log n$.*

Proof. Since two ω-regular languages are equivalent if they agree on the set of ultimately periodic words [17], it suffices to argue that A and A_k agree on ultimately periodic words.

From Lemma 11, A_k accepts an ultimately periodic word w if and only if Eve wins the k-register game on $\mathcal{G}(w, \mathcal{A})$. From Lemma 16, this is the case exactly when Eve wins the parity game on $\mathcal{G}(w, \mathcal{A})$, that is, when A accepts w. ◀

▶ **Theorem 18.** *The size blow-up and state blow-up involved in translating alternating parity word automata to alternating weak word automata is at most quasi-polynomial. In particular, every APW $\mathcal{A}$ of size (resp. number of states) n is equivalent to an AWW of size (resp. number of states) $2^{O((\log n)^3)}$.*

Proof. From Lemma 17, an APW $\mathcal{A}$ with n states and d priorities is equivalent to its parameterised APW $\mathcal{A}_k$ for $k = 1 + \log n$, having $n \cdot d^k \cdot (2k + 1)$ states and $2k + 1$ priorities. $\mathcal{A}_k$ can then be turned into a weak automaton using standard techniques [12] with a $O(m^{d'})$ blow-up, where m is the number of states and d' the number of priorities, which yields an AWW with $2^{O((\log n)^3)}$ many states, since m is here in $O(kn^{k+1}) \leq 2^{O((\log n)^2)}$ and d' is $2k + 1 \in O(\log n)$.

In case that the size of $\mathcal{A}$ is dominated by the size e of its transition function, namely when $e > n$, observe that the parameter k, the number of states in $\mathcal{A}_k$, and the number of priorities in $\mathcal{A}_k$ do not depend on e, while the size of $\mathcal{A}_k$'s transition function is in $O(k^2 e d^k) \leq 2^{O((\log e)^2)}$. Since the translation in [12] does not blow up the transition-function size more than it blows up the number of states, we end up with an AWW of size in $2^{O((\log e)^3)}$. ◀