Boundedness of Conjunctive Regular Path Queries

Pablo Barceló, Diego Figueira, Miguel Romero

Boundedness of Conjunctive Regular Path Queries

Pablo Barceló
Department of Computer Science, University of Chile & IMFD Chile
pbarcelo@dcc.uchile.cl

Diego Figueira
Univ. Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, F-33400, Talence, France
diego.figueira@labri.fr

Miguel Romero
University of Oxford, UK
miguel.romero@cs.ox.ac.uk

Abstract

We study the boundedness problem for unions of conjunctive regular path queries with inverses (UC2RPQs). This is the problem of, given a UC2RPQ, checking whether it is equivalent to a union of conjunctive queries (UCQ). We show the problem to be ExpSpace-complete, thus coinciding with the complexity of containment for UC2RPQs. As a corollary, when a UC2RPQ is bounded, it is equivalent to a UCQ of at most triple-exponential size, and in fact we show that this bound is optimal. We also study better behaved classes of UC2RPQs, namely *acyclic UC2RPQs of bounded thickness*, and *strongly connected UCRPQs*, whose boundedness problem is, respectively, PSpace-complete and Π_2^P-complete. Most upper bounds exploit results on limitedness for distance automata, in particular extending the model with alternation and two-wayness, which may be of independent interest.

2012 ACM Subject Classification Theory of computation → Database query languages (principles); Theory of computation → Quantitative automata

Keywords and phrases regular path queries, boundedness, limitedness, distance automata

Related Version A full version of this paper can be found at https://arxiv.org/abs/1904.00850 and https://hal.archives-ouvertes.fr/hal-02056388.

Funding Barceló is funded by the Millennium Inst. for Foundational Research on Data and Fondecyt 1170109, and Figueira by ANR project DELTA (grant ANR-16-CE40-0007) and ANR project QUID (grant ANR-18-CE40-0031). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 714532). The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

Acknowledgements We are grateful to Thomas Colcombet for helpful discussions and valuable ideas in relation to the results of Section 5.

1 Introduction

Boundedness is an important property of formulas in logics with fixed-point features. At the intuitive level, a formula φ in any such logic is bounded if its *fixed-point depth*, *i.e.*, the number of iterations that are needed to evaluate φ on a structure $\mathbf{A}$, is fixed (and thus it is independent of $\mathbf{A}$). In databases and knowledge representation, boundedness is regarded as an interesting theoretical phenomenon with relevant practical implications [25, 8]. In fact, while several applications in these areas require the use of recursive features, actual real-world systems are either not designed or not optimized to cope with the computational demands that such features impose. Bounded formulas, in turn, can be reformulated in

non-recursive logics, such as FO, or even as a *union of conjunctive queries* (UCQ) when φ itself is positive. UCQs form the core of most systems for data management and ontological query answering, and, in addition, are the focus of advanced optimization techniques. It has also been experimentally verified in some contexts that recursive features encountered in practice are often used in a somewhat 'harmless' way, and that many of such queries are in fact bounded [23]. Thus, checking if a recursive formula φ is bounded, and building an equivalent non-recursive formula φ' when the latter holds, are important optimization tasks.

The study of boundedness for *Datalog* programs, *i.e.*, the least fixed-point extension of the class of UCQs, received a lot of attention during the late 80s and early 90s. Two seminal results established that checking boundedness is undecidable in general for Datalog [22], but becomes decidable for *monadic* Datalog, *i.e.*, those programs in which each intensional predicate is monadic [19]. The past few years have seen a resurgence of interest in boundedness problems. This is due, in part, to the development of the theory of *cost automata* over trees (both finite and infinite) in a series of landmark results, in particular relating to its *limitedness* problem. In a few words, cost automata are generalizations of finite automata associating a *cost* from $\mathbb{N} \cup \{\infty\}$ to every input tree (instead of simply accepting or rejecting). The limitedness problem asks, given a cost automata, whether there is a uniform bound on the cost over all (accepting) input trees. Some deep results establish that checking limitedness is decidable for well-behaved classes of cost automata over trees [18, 36, 37, 7]. Remarkably, for several logics of interest the boundedness problem can be reduced to the limitedness for cost automata in such well-behaved classes. Those reductions have enabled powerful decidability results for the boundedness problem. As an example, it has been shown in this way that boundedness is decidable for monadic second-order logic (MSO) over structures of bounded treewidth [11], which corresponds to an extension of Courcelle's Theorem, and also for the *guarded negation* fragment of least fixed-point logic (LFP), even in the presence of unguarded parameters [6]. Cost automata have also been used to study the complexity of boundedness for guarded Datalog programs [7, 3].

Graph databases is a prominent area of study within database theory, in which the use of recursive queries is crucial [2, 1]. A graph database is a finite edge-labeled directed graph. The most basic navigational querying mechanism for graph databases corresponds to the class of *regular path queries* (RPQs), which check whether two nodes of the graph are connected by a path whose label belongs to a given regular language. RPQs are often extended with the ability to traverse edges in both directions, giving rise to the class of *two-way* RPQs, or 2RPQs [15]. The core of the most popular recursive query languages for graph databases is defined by *conjunctive* 2RPQs, or C2RPQs, which are the closure of 2RPQs under conjunction and existential quantifications [14]. We also consider *unions* of C2RPQs, or UC2RPQs. It can be shown that a UC2RPQ is bounded iff it is equivalent to some UCQ. In spite of the inherent recursive nature of UC2RPQs, their boundedness problem has not been studied in depth. Here we develop such a study by showing the following:

- The boundedness problem for UC2RPQs is ExpSpace-complete. The lower bound holds even for CRPQs. This implies that boundedness is not more difficult than *containment* for UC2RPQs, which was shown to be ExpSpace-complete in [14].
- From our upper bound construction it follows that if a UC2RPQ is bounded, then it is equivalent to a UCQ of triple exponential size. We show that this bound is optimal.
- Finally, we obtain better complexity bounds for some subclasses of UC2RPQs; namely, for acyclic UC2RPQs of *bounded thickness*, in which case boundedness becomes PSpace-complete, and for *strongly connected* UCRPQs, for which it is Π_2^P-complete.

It is important to stress that UC2RPQs can be easily translated into guarded LFP

with unguarded parameters, for which boundedness was shown to be decidable by applying sophisticated cost automata techniques as mentioned above. However, the complexity of the boundedness problem for such a logic is currently not well-understood – and it is at least 2ExpTime-hard [7] – and hence this translation does not yield, in principle, optimal complexity bounds for our problem. To study the boundedness for UC2RPQs, we develop instead techniques especially tailored to UC2RPQs. In fact, since the recursive structure of UC2RPQs is quite tame, their boundedness problem can be translated into the limitedness problem for a much simpler automata model than cost automata on trees; namely, *distance* automata on finite words. Distance automata are nothing more than usual NFAs with two sorts of transitions: costly and non-costly. Such an automaton is limited if there is an integer $k \geq 1$ such that every word accepted by the NFA has an accepting run with at most k costly transitions. A beautiful result in automata theory established the decidability of the limitedness problem for distance automata [24], which is actually in PSpace [30]. While being a difficult result, by now we have quite transparent proofs of this fact (see, *e.g.*, [26]). We exploit our translation to obtain tight complexity upper bounds for boundedness of UC2RPQs. Some of the proofs in the paper require extending the study of limitedness to *alternating* and *two-way* distance automata, while preserving the PSpace bound for the limitedness problem. We believe these results to be of independent interest.

Organization of the paper. Section 2 contains preliminaries. We present characterizations of boundedness for UC2RPQs in Section 3 and an application of those to pinpoint the complexity of Boundedness for RPQs in Section 4. Distance automata and results about them are given in Section 5. We analyze the complexity of Boundedness for general UC2RPQs in Section 6 and present some classes of UC2RPQs with better complexity of Boundedness in Section 7. We finish with a discussion in Section 8.

2 Preliminaries

We assume familiarity with *non-deterministic finite automata* (NFA), *two-way* NFA (2NFA), and *alternating finite automata* (AFA) over finite words. We often blur the distinction between an NFA $\mathcal{A}$ and the language $L(\mathcal{A})$ it defines; similarly for regular expressions.

Graph databases and conjunctive regular path queries. A *graph database* over a finite alphabet $\mathbb{A}$ is a finite edge-labelled graph $G = (V, E)$ over $\mathbb{A}$, where V is a finite set of vertices and $E \subseteq V \times \mathbb{A} \times V$ is the set of labelled edges. We write $u \xrightarrow{a} v$ to denote an edge $(u, a, v) \in E$. We define the alphabet $\mathbb{A}^{\pm} := \mathbb{A} \dot{\cup} \mathbb{A}^{-1}$ that extends $\mathbb{A}$ with the set $\mathbb{A}^{-1} := \{a^{-1} \mid a \in \mathbb{A}\}$ of "inverses" of symbols in $\mathbb{A}$. An *oriented path* from u to v in a graph database $G = (V, E)$ over alphabet $\mathbb{A}$ is a pair $\pi = (\sigma, \ell)$ where σ and ℓ are (possibly empty) sequences $\sigma = (v_0, a_1, v_1), (v_1, a_2, v_2), \ldots, (v_{k-1}, a_k, v_k) \in V \times \mathbb{A} \times V$, and $\ell = \ell_1, \ldots, \ell_k \in \{-1, 1\}$, for $k \geq 0$, such that $u = v_0$, $v = v_k$, and for each $1 \leq i \leq k$, we have that $\ell_i = 1$ implies $(v_{i-1}, a_i, v_i) \in E$; and $\ell_i = -1$ implies $(v_i, a_i, v_{i-1}) \in E$. The *label* of π is the word $b_1 \ldots b_k \in (\mathbb{A}^{\pm})^*$, where $b_i = a_i$ if $\ell_i = 1$; otherwise $b_i = a_i^{-1}$. When $k = 0$ the label of π is the empty word ε. If $\ell_i = 1$ for every $1 \leq i \leq k$, we say that π is a *directed path*. Note that in this case, the label of π belongs to $\mathbb{A}^*$.

A *regular path query* (RPQ) over $\mathbb{A}$ is a regular language $L \subseteq \mathbb{A}^*$, which we assume to be given as an NFA. The evaluation of L on a graph database $G = (V, E)$ over $\mathbb{A}$, written $L(G)$, is the set of pairs $(u, v) \in V \times V$ such that there is a directed path from u to v in G whose label belongs to L. 2RPQs extend RPQs with the ability to traverse edges in both

directions. Formally, a 2RPQ L over $\mathbb{A}$ is simply an RPQ over $\mathbb{A}^{\pm}$. The evaluation $L(G)$ of L over a graph database $G = (V, E)$ over $\mathbb{A}$ is the set of pairs $(u, v) \in V \times V$ such that there is an oriented path from u to v in G whose label belongs to L.

Conjunctive 2RPQs (C2RPQs) are obtained by taking the closure of 2RPQs under conjunction and existential quantification, *i.e.*, a C2RPQ over $\mathbb{A}$ is an expression $\gamma := \exists \bar{z} \left((x_1 \xrightarrow{L_1} y_1) \wedge \cdots \wedge (x_m \xrightarrow{L_m} y_m) \right)$, where each L_i is a 2RPQ over $\mathbb{A}$ and $\bar{z}$ is a tuple of variables among those in $\{x_1, y_1, \ldots, x_m, y_m\}$. We say that γ is a CRPQ if each L_i is an RPQ. If $\bar{x} = (x^1, \ldots, x^n)$ is the tuple of *free variables* of γ, *i.e.*, those that are not existentially quantified in $\bar{z}$, then the evaluation $\gamma(G)$ of the C2RPQ γ over a graph database G is the set of all tuples $h(\bar{x}) = (h(x^1) \ldots, h(x^n))$, where h ranges over all mappings $h : \{x_1, y_1, \ldots, x_m, y_m\} \to V$ such that $(h(x_i), h(y_i)) \in L_i(G)$ for each $1 \leq i \leq m$.

A *union* of C2RPQs (UC2RPQ) is an expression of the form $\Gamma := \bigvee_{1 \leq i \leq n} \gamma_i$, where the γ_i's are C2RPQ, all of which have exactly the same free variables. The evaluation $\Gamma(G)$ of Γ over a graph database G is $\bigcup_{1 \leq i \leq n} \gamma_i(G)$. We often write $\Gamma(\bar{x})$ to denote that $\bar{x}$ is the tuple of free variables of Γ. A UC2RPQ Γ is *Boolean* if it contains no free variables.

Given UC2RPQs Γ and Γ', we write $\Gamma \subseteq \Gamma'$ if $\Gamma(G) \subseteq \Gamma'(G)$ for each graph database G. Hence, Γ and Γ' are *equivalent* if $\Gamma \subseteq \Gamma'$ and $\Gamma' \subseteq \Gamma$, *i.e.*, $\Gamma(G) = \Gamma'(G)$ for every G.

Boundedness of UC2RPQs. CRPQs, and even UC2RPQs, can easily be expressed in *Datalog*, the least fixed-point extension of the class of *union of conjunctive queries* (UCQs). Hence, we can directly define the boundedness of a UC2RPQ in terms of the boundedness of its equivalent Datalog program, which is a well-studied problem [25]. The latter, however, coincides with being equivalent to some UCQ [32]. In the setting of graph databases, a *conjunctive query* (CQ) over $\mathbb{A}$ is simply a CRPQ over $\mathbb{A}$ of the form $\exists \bar{z} \bigwedge_{1 \leq i \leq m} (x_i \xrightarrow{a_i} y_i)$ where the a_is range over $\mathbb{A} \cup \{\varepsilon\}$. Notice that atoms of the form $x \xrightarrow{\varepsilon} y$ correspond to *equality atoms* $x = y$. Analogously, one can define unions of CQs (UCQs). Note that, modulo equality atoms, a CQ over $\mathbb{A}$ can be seen as a graph database over $\mathbb{A}$. Hence, we shall slightly abuse notation and, in the setting of CQs, use notions defined for graph databases (such as oriented paths).

A UC2RPQ Γ is *bounded* if it is equivalent to some UCQ Φ. In this article we study the complexity of the problem Boundedness, which takes as input a UC2RPQ Γ and asks whether Γ is bounded.

▶ **Example 1.** Consider the Boolean UCRPQ $\Gamma = \gamma_1 \vee \gamma_2$ over the alphabet $\mathbb{A} = \{a, b, c, d\}$ such that $\gamma_1 = \exists x, y \, (x \xrightarrow{L_b} y \wedge x \xrightarrow{L_{b,d}} y)$ and $\gamma_2 = \exists x, y \, (x \xrightarrow{L_d} y \wedge x \xrightarrow{L_{b,d}} y)$, where $L_b := a^+b^+c$, $L_d := ad^+c^+$, and $L_{b,d} := a^+(b + d)c^+$. For $e \in \mathbb{A}$, recall that e^+ denotes the language $e(e^*)$. As we shall explain in Example 4, we have that γ_1 and γ_2 are unbounded. However, Γ is bounded, and in particular, it is equivalent to the UCQ $\Phi = \varphi_1 \vee \varphi_2$, where φ_1 and φ_2 correspond to $\exists x, y \, (x \xrightarrow{abc} y)$ and $\exists x, y \, (x \xrightarrow{adc} y)$, respectively. ◀

3 Characterizations of Boundedness for UC2RPQs

In this section we provide two simple characterizations of when a UC2RPQ is bounded that will be useful to analyze the complexity of Boundedness. Let $\varphi(\bar{x})$ and $\varphi'(\bar{x})$ be CQs over $\mathbb{A}$ with variable sets $\mathcal{V}$ and $\mathcal{V}'$, respectively. Let $=_\varphi$ and $=_{\varphi'}$ be the binary relations induced on $\mathcal{V}$ and $\mathcal{V}'$ by the equality atoms of φ and φ', respectively, and $=^*_\varphi$ and $=^*_{\varphi'}$ be their reflexive-transitive closure. A *homomorphism* from φ to φ' is a mapping $h : \mathcal{V} \to \mathcal{V}'$ such that: (i) $x =^*_\varphi y$ implies $h(x) =^*_{\varphi'} h(y)$; (ii) $h(\bar{x}) = \bar{x}$; and (iii) for each atom $x \xrightarrow{a} y$

in φ with $a \in \mathbb{A}$, there is an atom $x' \xrightarrow{a} y'$ in φ' such that $h(x) =^*_{\varphi'} x'$ and $h(y) =^*_{\varphi'} y'$. We write $\varphi \to \varphi'$ if such a homomorphism exists. It is known that $\varphi \to \varphi'$ iff $\varphi' \subseteq \varphi$ [16].

An *expansion* of a C2RPQ $\gamma(\bar{x})$ over $\mathbb{A}$ is a CQ $\lambda(\bar{x})$ over $\mathbb{A}$ with minimal number of variables and atoms such that (i) λ contains each variable of γ, (ii) for each atom $A = x \xrightarrow{L} y$ of γ, there is an oriented path π_A in λ from x to y with label $w_A \in L$ whose intermediate variables (*i.e.*, those not in $\{x, y\}$) are distinct from one another, and (iii) intermediate variables of different oriented paths π_A and $\pi_{A'}$ are disjoint. Note that the free variables of λ and γ coincide. Intuitively, the expansion λ is obtained from γ by choosing for each atom $A = x \xrightarrow{L} y$ a word $w_A \in L$, and "expanding" $x \xrightarrow{L} y$ into the "fresh oriented path" π_A from x to y with label w_A. When $w_A = \varepsilon$ then λ contains the equality atom $x = y$. An expansion of a UC2RPQ Γ is an expansion of some C2RPQ in Γ. Observe that a (U)C2RPQ is always equivalent to the (potentially infinite) UCQ given by its set of expansions. Even more, it is equivalent to the UCQ defined by its *minimal* expansions, as introduced below.

If λ is an expansion of a UC2RPQ Γ, we define the *size* of λ, denoted by $\|\lambda\|$, to be the number of (non-equality) atoms in λ. We say that λ is *minimal*, if there is no expansion λ' such that $\lambda' \to \lambda$ and $\|\lambda'\| < \|\lambda\|$. Intuitively, an expansion is minimal if its answers cannot be covered by a smaller expansion. We can then establish the following.

▶ **Lemma 2.** *Every UC2RPQ Γ is equivalent to the (potentially infinite) UCQ given by its set of minimal expansions.*

We can now provide our basic characterizations of boundedness.

▶ **Proposition 3.** *The following conditions are equivalent for each UC2RPQ Γ.*

1. *Γ is bounded.*
2. *There is $k \geq 1$ such that for every expansion λ of Γ there exists an expansion λ' of Γ with $\|\lambda'\| \leq k$ such that $\lambda \subseteq \lambda'$ (i.e., such that $\lambda' \to \lambda$).*
3. *Γ has finitely many minimal expansions.*

▶ **Example 4.** Consider the Boolean UCRPQ $\Gamma = \gamma_1 \vee \gamma_2$ over $\mathbb{A} = \{a, b, c, d\}$ from Example 1. To see that γ_1 is unbounded (the case of γ_2 is similar) we can apply Proposition 3. Indeed, the expansions of γ_1 corresponding to $\{\exists x, y\,(x \xrightarrow{ab^n c} y \wedge x \xrightarrow{adc} y) : n \geq 1\}$ are all minimal. On the other hand, Γ is bounded as its minimal expansions correspond to $\exists x, y\,(x \xrightarrow{abc} y \wedge x \xrightarrow{abc} y)$ and $\exists x, y\,(x \xrightarrow{adc} y \wedge x \xrightarrow{adc} y)$. ◀

4 Boundedness for Existentially Quantified RPQs

As a first application of Proposition 3, we study Boundedness for CRPQs consisting of a single RPQ; that is, RPQs or existentially quantified RPQs. Let v, w be words over $\mathbb{A}$. Recall that a word v is a *prefix* [resp. *suffix* and *factor*] of w if $w \in v \cdot \mathbb{A}^*$ [resp. $w \in \mathbb{A}^* \cdot v$ and $w \in \mathbb{A}^* \cdot v \cdot \mathbb{A}^*$]. If in addition we have $v \neq w$, then we say that v is a *proper* prefix [resp. suffix and factor] of w. For a language $L \subseteq \mathbb{A}^*$, we define its *prefix-free* sub-language L_{pf} to be the set of words $w \in L$ such that w has no proper prefix in L. Similarly, we define L_{sf} and L_{ff} with respect to the suffix and factor relation. We have the following:

▶ **Proposition 5.** *The following statements hold.*

1. *An RPQ L is bounded iff L is finite.*
2. *A CRPQ $\exists y(x \xrightarrow{L} y)$ [resp. $\exists x(x \xrightarrow{L} y)$] with $x \neq y$ is bounded iff L_{pf} [resp. L_{sf}] is finite.*
3. *A Boolean CRPQ $\exists x, y(x \xrightarrow{L} y)$ with $x \neq y$ is bounded iff L_{ff} is finite.*

▶ **Theorem 6.** *The problem of, given an NFA accepting the language L, checking whether L_{pf} is finite is* PSPACE*-complete. The same holds if we replace L_{pf} by L_{sf} or L_{ff}.*

Proof. We focus on upper bounds, the lower bounds are in the appendix. Given an NFA $\mathcal{A}$ accepting the language L, we can construct an NFA $\mathcal{B}$ of polynomial size in $\mathcal{A}$ that accepts precisely those words that have a proper prefix in L. By complementing and intersecting with $\mathcal{A}$, we obtain an NFA $\mathcal{B}'$ of exponential size in $\mathcal{A}$ that accepts the language L_{pf}. Hence, we only need to check whether the language accepted by $\mathcal{B}'$ is finite, which can be done *on-the-fly* in NL w.r.t. $\mathcal{B}'$, and hence in PSPACE. The other two cases are analogous. ◀

By applying Theorem 6 and Proposition 5, we can now pinpoint the complexity of BOUNDEDNESS for CRPQs with a single RPQ.

▶ **Corollary 7.** *The following statements hold.*

1. BOUNDEDNESS *for RPQs is* NL*-complete.*
2. BOUNDEDNESS *for CRPQs of the form $\exists y(x \xrightarrow{L} y)$, with $x \neq y$, is* PSPACE*-complete. The same holds for CRPQs $\exists x(x \xrightarrow{L} y)$ and Boolean CRPQs $\exists x, y(x \xrightarrow{L} y)$, where $x \neq y$.*

It is not clear, though, how usual automata techniques, as the ones applied in the proof of Theorem 6, can be used to solve BOUNDEDNESS for more complex CRPQs. To solve this problem we develop an approach based on distance automata, as introduced next. Our approach also handles inverses and unions, thus dealing with arbitrary UC2RPQs.

5 Distance Automata

Distance automata [24] (equivalent to weighted automata over the $(\min, +)$-semiring [21], min-automata [12], or $\{\varepsilon, ic\}$-B-automata [17]) are an extension of finite automata which associate to each word in the language a natural number or 'cost'. They can be represented as non-deterministic finite automata with two sorts of transitions: costly and non-costly. For a given distance automaton, the cost of a run on a word is the number of costly transitions, and the cost of a word $w \in \mathbb{A}^*$ is the minimum cost of an accepting run on w. We will use this automaton model to encode boundedness as the problem of whether there is a uniform bound on the cost of words, known as the *limitedness problem*.

Formally, a *distance automaton* (henceforth *DA*) is a tuple $\mathcal{A} = (\mathbb{A}, Q, q_0, F, \delta)$, where $\mathbb{A}$ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of finals states and $\delta \subseteq Q \times \mathbb{A} \times \{0, 1\} \times Q$ is the transition relation. A word $w \in \mathbb{A}^*$ is *accepted* by $\mathcal{A}$ if there is an *accepting run* of $\mathcal{A}$ on w, *i.e.*, a (possibly empty) sequence of transitions $\rho = (p_1, a_1, c_1, r_1) \cdots (p_n, a_n, c_n, r_n) \in \delta^*$ with the usual properties: (1) if $\rho = \varepsilon$ then $q_0 \in F$ and $w = \varepsilon$, (2) $p_1 = q_0$ and $r_n \in F$, (3) for every $1 \leq i < n$ we have $r_i = p_{i+1}$, and (4) $w = a_1 \cdots a_n$. The *cost* of the run ρ is $cost(\rho) = c_1 + \cdots + c_n$ (or 0 if $\rho = \varepsilon$); and the cost $cost_{\mathcal{A}}(w)$ of a word w accepted by $\mathcal{A}$ is the minimum cost of an accepting run of $\mathcal{A}$ on w. For convenience, we assume the cost of words not accepted by $\mathcal{A}$ to be 0.

The *limitedness problem* for DA is defined as follows: given a DA $\mathcal{A}$, determine whether $\sup_{w \in \mathbb{A}^*} cost_{\mathcal{A}}(w) < \infty$. This problem is known to be PSPACE-complete.

▶ **Theorem 8.** *[29, 30] The following statements hold:*

1. *The limitedness problem for DA is* PSPACE*-complete.*
2. *If a DA with n states is limited, then $\sup_{w \in \mathbb{A}^*} cost_{\mathcal{A}}(w) \leq 2^{O(n^3)}$.*

We use two extensions of DA: *alternating* and *two-way*. Two-way DA is defined as for NFA, extending the cost function accordingly. The cost of a word is still the minimum over

the cost of all (potentially infinitely many) runs. Alternating DA is defined as usual by having two sorts of states: universal and existential. Existential states can be seen as computing the minimum among the cost of all possible continuations of the run, and universal states as computing the maximum (or supremum if the automaton is also two-way). As we will see, these extensions preserve the above PSPACE upper bound for the limitedness problem.

Formally, an *alternating two-way DA with epsilon transitions* (A2DA$^\varepsilon$) over $\mathbb{A}$ is a tuple $\mathcal{A} = (\mathbb{A}, Q_\exists, Q_\forall, q_0, F, \delta)$ is an A2DA$^\varepsilon$ if $q_0 \in Q_\exists$, $F \subseteq Q_\exists$ and

$$\delta \subseteq (Q_\exists \cup Q_\forall) \times (\mathbb{A}^\pm \cup \{\varepsilon\}) \times \{end, \overline{end}\} \times \{0,1\} \times (Q_\exists \cup Q_\forall);$$

where *end* indicates that after reading the letter we arrive at the end of the word (*i.e.*, either the leftmost or the rightmost end) and $\overline{end}$ indicates that we do not. When the automaton $\mathcal{A}$ is two-way, it is convenient to think of its head as being *between* the letter positions of the word, so an *end*-flagged transition can be applied only if it moves the head to be right before the first letter of the word, or right after the last one.

For any given word $w \in \mathbb{A}^*$, consider the edge-labelled graph $G_{\mathcal{A},w} = (V, E)$ over δ, where $V = Q \times \{0, \ldots, |w|\}$, with $Q = Q_\exists \cup Q_\forall$, and $E \subseteq V \times \delta \times V$ consists of all edges $(q,i) \xrightarrow{(q,a,e,c,p)} (p,j)$ such that $e = end$ iff $j = 0$ or $j = |w|$ and either (a) $i < |w|$, $a = w[i+1]$, and $j = i+1$; (b) $i > 0$, $a = (w[i])^{-1}$, and $j = i-1$; or (c) $a = \varepsilon$ and $j = i$.

An *accepting run of $\mathcal{A}$ on w from* $(q,i) \in Q \times \{0, \ldots, |w|\}$ is a finite (possibly empty) edge-labelled directed rooted tree[1] t over δ and a labelling h from the nodes of t to the nodes of $G_{\mathcal{A},w}$, such that if t is empty then $q \in F$, and otherwise h maps the root of t to (q,i), every leaf of t to $F \times \{0, \ldots, |w|\}$, and for every node x of t:

- if (x, α, y) is an (labeled) edge in t for some y, then $(h(x), \alpha, h(y))$ is an edge in $G_{\mathcal{A},w}$;
- if $h(x) \in Q_\forall \times \{0, \ldots, |w|\}$, then for every edge $(h(x), \alpha, c)$ in $G_{\mathcal{A},w}$, there is an edge (x, α, y) in t so that $h(y) = c$;
- if $h(x) \in Q_\exists \times \{0, \ldots, |w|\}$, then x has at most one child.

Each branch of t with label $(q_1, a_1, e_1, c_1, p_1), \ldots, (q_n, a_n, e_n, c_n, p_n)$ has an associated cost of $c_1 + \cdots + c_n$; and the cost associated with t is the maximum among the costs of its branches, or 0 if t is empty. The cost $cost_\mathcal{A}(w, q, i)$ is the minimum cost of an accepting run on w from (q,i), or 0 if none exists; $cost_\mathcal{A}(w)$ is defined as $cost_\mathcal{A}(w, q_0, 0)$.

An A2DA$^\varepsilon$ with $\delta \subseteq Q \times (\mathbb{A} \cup \{\varepsilon\}) \times \{end, \overline{end}\} \times \{0,1\} \times Q$ is an *alternating DA* with ε transitions (ADA$^\varepsilon$). An A2DA$^\varepsilon$ with $Q_\forall = \emptyset$ is a *two-way DA* with ε transitions (2DA$^\varepsilon$). An A2DA with both the aforementioned conditions is (equivalent to) a DA with ε transitions (DA$^\varepsilon$). Notice that in the last two cases, accepting runs can be represented as words from δ^* rather than trees. By A2DA (resp., ADA, 2DA, DA) we denote an A2DA$^\varepsilon$ (resp., ADA$^\varepsilon$, 2DA$^\varepsilon$, DA$^\varepsilon$) with no ε-transitions. Note that DA as just defined is in every sense equivalent to the distance automata model we have defined at the beginning of this section —this is why we overload the same 'DA' name.

We first observe that 2DA can be transformed into DA while preserving both the language and limitedness problems by adapting the standard "crossing sequence" construction for translating 2NFA into NFA [35]. This fact will be useful for proving the EXPSPACE upper bound for BOUNDEDNESS of general UC2RPQs in Section 6.

▶ **Proposition 9.** *There is an exponential time procedure which for every 2DA $\mathcal{A}$ over $\mathbb{A}$ produces a DA $\mathcal{B}$ over $\mathbb{A}$ such that the languages accepted by $\mathcal{A}$ and $\mathcal{B}$ are the same, and*

[1] That is, a tree-shaped finite edge-labelled graph over δ with edges directed in the root-to-leaf sense.

$cost_{\mathcal{B}}(w) \leq cost_{\mathcal{A}}(w) \leq f(cost_{\mathcal{B}}(w))$ for every $w \in \mathbb{A}^*$, where f is a polynomial function that depends on the number of states of $\mathcal{A}$.

Recall that the universality problem for NFAs is known to be PSPACE-complete [27]; and that this bound actually extends to two-way and even alternating automata. We show that, likewise, the limitedness problem remains in PSPACE for A2DA$^{\varepsilon}$. This result will be useful to show in Section 7 that BOUNDEDNESS for the class of acyclic UC2RPQs of bounded thickness is in PSPACE.

▶ **Theorem 10.** *The limitedness problem for A2DA$^{\varepsilon}$ is* PSPACE*-complete.*

The novelty of this result is the PSPACE upper bound. In fact, decidability follows from known results, and in particular [7, Theorem 14] claims EXPTIME-membership in the more challenging setup of infinite trees. However, this is obtained via an involved construction spanning through several papers. The proof of Theorem 10, instead, is obtained by the composition of the following reductions:

$$\text{lim. A2DA}^{\varepsilon} \xrightarrow{(1)} \text{lim. A2DA} \xrightarrow{(2)} \text{lim. 2DA} \xrightarrow{(3)} \text{lim. ADA}^{\varepsilon} \xrightarrow{(4)} \text{lim. ADA} \xrightarrow{(5)} \text{lim. DA}.$$

Reductions (1), (3) and (4) are in polynomial time, while reductions (2) and (5), which are basically the same, are in exponential time. Specifically, reductions (2) and (5) preserve the statespace but the size of the alphabet grows exponentially in the number of states and linearly in the size of the source alphabet. However, the alphabet and transition set resulting from these reductions can be succinctly described: letters are encoded in polynomial space, and checking for membership in the transition set is polynomial time computable.

In summary, the composition (1)+(2)+(3)+(4)+(5) yields a DA with the following characteristics: (i) it has a polynomial number of states Q; (ii) it runs on an exponential alphabet $\mathbb{A}$ –and every letter is encoded in polynomial space–; and (iii) one can check in polynomial time whether a tuple $t \in Q \times \mathbb{A} \times \{end, \overline{end}\} \times \{0,1\} \times Q$ is in its transition relation. This, coupled with Theorem 8, item (2) (which offers a bound depending only on the number of states), provides a polynomial space algorithm for the limitedness of A2DA$^{\varepsilon}$: We can non-deterministically check the existence of a word with cost greater than the single exponential bound N using only polynomial space, by guessing one letter at a time and keeping the set of reachable states together with the associated costs, where each cost is encoded in binary using polynomial space if it is smaller than N, or with a '∞' flag otherwise. The algorithm accepts if at least one final state is reached and the costs of all reachable final states are marked ∞. Since NPSPACE =PSPACE (Savitch's Theorem), Theorem 10 follows.

We now provide a brief description of the reductions used in the proof of Theorem 10.

(1) From A2DA$^{\varepsilon}$ to A2DA This is a trivial reduction obtained by simulating ε-transitions by reading $a \cdot a^{-1}$ for some $a \in \mathbb{A}$.

(2) From A2DA to 2DA Given an A2DA $\mathcal{A} = (\mathbb{A}, Q_\forall, Q_\exists, q_0, F, \delta)$, we build a 2DA $\mathcal{B}$ over a larger alphabet $\mathbb{B}$, where we trade alternation for extra alphabet letters. The alphabet $\mathbb{B}$ consists of triples $(f^{\rightarrow}, a, f^{\leftarrow})$, where $a \in \mathbb{A}$ and $f^{\rightarrow}, f^{\leftarrow} : Q_\forall \to \delta$. The idea is that $f^{\rightarrow}, f^{\leftarrow}$ are "choice functions" for the alternation: whenever we are to the left (resp., right) of a position of the word labelled $(f^{\rightarrow}, a, f^{\leftarrow})$ in state $q \in Q_\forall$, instead of exploring all transitions departing from q and taking the maximum cost over all such runs (this is what alternation does in $\mathcal{A}$), $\mathcal{B}$ chooses to just take the transition $f^{\rightarrow}(q)$ (resp., $f^{\leftarrow}(q)$). Note that $\mathbb{B}$ is exponential in the number of states but not in the size of $\mathbb{A}$. In this way, we build a 2DA $\mathcal{B}$ having the same set of states as $\mathcal{A}$ but with a transition function which is essentially deterministic on the states of $Q_\forall$. In the end we obtain that

- for every $w \in \mathbb{B}^*$, $cost_{\mathcal{B}}(w) \leq cost_{\mathcal{A}}(w_{\mathbb{A}})$; and
- for every $w \in \mathbb{A}^*$ there is $\widetilde{w} \in \mathbb{B}^*$ so that $\widetilde{w}_{\mathbb{A}} = w$ and $cost_{\mathcal{A}}(w) = cost_{\mathcal{B}}(\widetilde{w})$,

where $w_{\mathbb{A}}$ and $\widetilde{w}_{\mathbb{A}}$ denote the projections onto the alphabet $\mathbb{A}$. This implies that the limitedness problem is preserved.

(3) From 2DA to ADA$^\varepsilon$ We show a polynomial-time translation from 2DA to ADA$^\varepsilon$ which preserves limitedness. In the case of finite automata, there are language-preserving reductions from 2NFA to AFA with a quadratic blowup in the statespace [9, 33]. However, these translations, when applied blindly to reduce from 2DA to ADA$^\varepsilon$, preserve neither the cost semantics nor the limitedness of languages. On the other hand, [10] shows an involved construction that results in a reduction from 2DA to ADA$^\varepsilon$ on *infinite trees*, which preserves limitedness but it is not polynomial in the number of states. We show a translation from 2DA to ADA$^\varepsilon$ which serves our purpose: it preserves limitedness and it is polynomial time computable. The translation is close to the language-preserving reduction from 2NFA to AFA of [33], upgraded to take into account the cost of different alternation branches, somewhat in the same spirit as the *history summaries* from [10].

(4) From ADA$^\varepsilon$ to ADA This is a straightforward polynomial time reduction which preserves limitedness but —as opposed to (1)— does not preserve the language: we need to add an extra letter to the alphabet in order to make the reduction work in polynomial time.

(5) From ADA to DA This is exactly the same reduction as (2), noticing that the alphabet will still be single exponential in the original A2DA$^\varepsilon$.

6 Complexity of Boundedness for UC2RPQs

Here we show that Boundedness for UC2RPQs is ExpSpace-complete. We do so by applying distance automata results presented in the previous section on top of the semantic characterizations presented in Section 3. The lower bound applies even for CRPQs. We further show that there is a triple exponential tight bound for the size of the equivalent UCQ of a UC2RPQ (and even CRPQ), whenever this exists. This is summarized in the following theorem. If Γ is a UC2RPQ, we write $\|\Gamma\|$ for the length of an arbitrary reasonable encoding of Γ —in particular, encodings in which regular languages are described through NFA or regular expressions.

▶ **Theorem 11.** *The following statements hold.*

1. Boundedness *for UC2RPQs is* ExpSpace*-complete. The problem remains* ExpSpace*-hard even for Boolean CRPQs.*
2. *If a UC2RPQ Γ is bounded, there is a UCQ Φ that is equivalent to Γ and such that Φ has at most triple-exponentially many CQs, each one of which is at most of double exponential size with respect to $\|\Gamma\|$.*
3. *There is a family $\{\Gamma_n\}_{n\geq 1}$ of Boolean CRPQs such that for each $n \geq 1$ it is the case that: (1) $\|\Gamma_n\| = O(n)$, (2) Γ_n is bounded, and (3) every UCQ that is equivalent to Γ_n has at least triple-exponentially many CQs with respect to n.*

6.1 Upper bounds

Our upper bound proof builds on top of techniques developed by Calvanese et al. [14] for studying the *containment problem for UC2RPQs*: Given UC2RPQs Γ, Γ', is it the case that $\Gamma \subseteq \Gamma'$? It is shown in [14] that from Γ, Γ' it is possible to construct exponentially sized NFAs $\mathcal{A}_{\Gamma,\Gamma'}$ and $\mathcal{A}'_{\Gamma,\Gamma'}$, such that $\Gamma \subseteq \Gamma'$ iff there is a word in $\mathcal{A}_{\Gamma,\Gamma'} \cap \overline{\mathcal{A}'_{\Gamma,\Gamma'}}$. It is a

well-known result that the latter is solvable in NL in the combined size of $(\mathcal{A}_{\Gamma,\Gamma'}, \overline{\mathcal{A}'_{\Gamma,\Gamma'}})$, *i.e.*, in ExpSpace. We modify this construction to study the boundedness of a given UC2RPQ Γ. In particular, we construct from Γ in exponential time a DA $\mathcal{D}_\Gamma$ such that Γ is bounded iff $\mathcal{D}_\Gamma$ is limited. The result then follows from Theorem 8, which establishes that limitedness for $\mathcal{D}_\Gamma$ can be solved in polynomial space on the number of its states, and thus in ExpSpace.

▶ **Proposition 12.** *There is a single exponential time procedure that takes as input a UC2RPQ Γ and constructs a DA $\mathcal{D}_\Gamma$ such that Γ is bounded iff $\mathcal{D}_\Gamma$ is limited.*

Proof. Similarly as done in [14], the DA $\mathcal{D}_\Gamma$ will run over encodings of expansions of the UC2RPQ Γ, *i.e.*, words over the alphabet $\mathbb{A}_1 := \mathbb{A}^\pm \cup \mathcal{V} \cup \{\$\}$, where $\mathbb{A}$ is the alphabet of Γ, $\mathcal{V}$ is the set of variables of Γ, and \$ is a fresh symbol. If $\gamma = \exists \bar{z} \bigwedge_{1 \leq i \leq m} (x_i \xrightarrow{L_i} y_i)$ is a C2RPQ in Γ and λ is the expansion of γ obtained by expanding each $x_i \xrightarrow{L_i} y_i$ into an oriented path π_i from x_i to y_i with label $w_i \in L_i$, then we encode λ as the word

$$w_\lambda = \$x_1 w_1 y_1 \$ x_2 w_2 y_2 \$ \cdots \$ x_m w_m y_m \$ \ \in \ \mathbb{A}_1^*$$

Note how the subword $x_i w_i y_i$ encodes the oriented path π_i. Every position $j \in \{1, \ldots, |w_\lambda|\}$ with $w_\lambda[j] \neq \$$ represents a variable in λ: either x_i or y_i if $w_\lambda[j] = x_i$ or $w_\lambda[j] = y_i$, respectively; or the $(\ell+1)$-th variable in the oriented path π_i if $w_\lambda[j]$ is the ℓ-th symbol in the subword w_i. Hence different positions in w_λ could represent the same variable in λ, *e.g.*, in the encoding $\$xabcy\$$, the $5th$ position containing a 'c' and the $6th$ position containing a 'y', represent the same variable, namely, the last vertex y of the oriented path. It is then easy to build, in polynomial time, an NFA $\mathcal{A}_1$ over $\mathbb{A}_1$ recognizing the language of all such encodings of expansions of Γ. Our automaton $\mathcal{D}_\Gamma$ is the product of $\mathcal{A}_1$ and the DA $\mathcal{C}_\Gamma$ defined below. In particular, $\mathcal{D}_\Gamma$ is limited iff $\mathcal{C}_\Gamma$ is limited over words of the form w_λ, for λ an expansion of Γ.

Fix a disjunct γ of Γ. As in [14], we consider words over the alphabet $\mathbb{A}_2 := \mathbb{A}_1 \times (2^\mathcal{V} \cup \{\#\})$ of the form $(\ell_1, \alpha_1) \cdots (\ell_n, \alpha_n)$, such that $w_\lambda = \ell_1 \cdots \ell_n$, for some expansion λ of Γ, and the α_i's are *valid γ-annotations*, *i.e.*, (1) $\alpha_i = \#$ if $\ell_i = \$$, (2) $\alpha_1, \ldots, \alpha_n \in 2^\mathcal{V}$ induce a partition of the variable set $\mathcal{V}_\gamma$ of γ, and (3) for each free variable $x \in \mathcal{V}_\gamma$ there is some (ℓ_i, α_i) such that $\ell_i = x$ and $x \in \alpha_i$. It is easy to construct an NFA $\mathcal{B}_1^\gamma$ of exponential size that given $w = (\ell_1, \alpha_1) \cdots (\ell_n, \alpha_n)$ with $w_\lambda = \ell_1 \cdots \ell_n$, checks if the α_i's are valid γ-annotations. Note that if the latter holds, then the annotations encode a mapping h_w from $\mathcal{V}_\gamma$ to the variables of λ such that $h_w(\bar{x}) = \bar{x}$, where $\bar{x}$ are the free variables of γ.

Now, given $w = (\ell_1, \alpha_1)(\ell_2, \alpha_2) \cdots (\ell_n, \alpha_n)$ with $w_\lambda = \ell_1 \cdots \ell_n$ and the α_i's being valid γ-annotations, it is shown in [14] that one can construct in polynomial time a 2NFA $\mathcal{B}_2^\gamma$ that checks the existence of an expansion λ' of γ and a homomorphism h from λ' to λ consistent with h_w. For each atom $x \xrightarrow{L} y$ of γ, the automaton $\mathcal{B}_2^\gamma$ guesses an oriented path π in λ from $h_w(x)$ to $h_w(y)$ with label $w' \in L$, directly over the encoding w_λ starting at a position j_x and ending at a position j_y in $\{0, \ldots, n\}$ (recall that the head moves in $\{0, \ldots, n\}$) with $j_x, j_y > 0$, $w[j_x] = (\ell, \alpha)$, $w[j_y] = (\ell', \alpha')$, $x \in \alpha$ and $y \in \alpha'$. Note that we have two types of transitions: (1) transitions that consume $a \in \mathbb{A}^\pm$ and actually guess an atom of π, and (2) transitions to "jump" from position j to j' in $\{0, \ldots, n\}$ representing *equivalent* variables of λ. The latter means that $j, j' > 0$ and either $w_\lambda[j]$ and $w_\lambda[j']$ represents exactly the *same* variable of λ, or $w_\lambda[j]$ and $w_\lambda[j']$ represent variables z, z' of λ such that $z =^*_\lambda z'$, where $=^*_\lambda$ is the reflexive-transitive closure of the relation induced by the equality atoms in λ.

Let $\mathcal{D}_2^\gamma$ be the 2DA obtained from the 2NFA $\mathcal{B}_2^\gamma$ by setting to 0 and 1 the cost of transitions of type (2) and (1), respectively. Hence, for a word w such that the projection of w to $\mathbb{A}_1$ is w_λ, and the one to $(2^\mathcal{V} \cup \{\#\})$ is a valid γ-annotation, we have that $cost_{\mathcal{D}_2^\gamma}(w)$ is precisely the

minimum size of an expansion λ' that can be mapped to λ via a homomorphism compatible with h_w. By Proposition 9, we can construct in exponential time in $\mathcal{D}_2^\gamma$ a DA $\mathcal{C}_2^\gamma$ accepting the same language as $\mathcal{D}_2^\gamma$ and having an exponential number of states, so that for every word w', we have $cost_{\mathcal{C}_2^\gamma}(w') \leq cost_{\mathcal{D}_2^\gamma}(w') \leq f(cost_{\mathcal{C}_2^\gamma}(w'))$ for some polynomial function f. Let $\exists\mathcal{C}^\gamma$ be the result of taking the product of $\mathcal{B}_1^\gamma$ and $\mathcal{C}_2^\gamma$ and then projecting over the alphabet $\mathbb{A}_1$. For every expansion λ of Γ, if λ' is a minimal size expansion of γ such that $\lambda' \to \lambda$, then we obtain that $cost_{\exists\mathcal{C}^\gamma}(w_\lambda) \leq \|\lambda'\| \leq f(cost_{\exists\mathcal{C}^\gamma}(w_\lambda))$. We define our desired $\mathcal{C}_\Gamma$ to be the union of $\exists\mathcal{C}^\gamma$ over all γ in Γ. We have that for every expansion λ, if λ_{min} is a minimal size expansion of Γ such that $\lambda_{min} \to \lambda$, then $cost_{\mathcal{C}_\Gamma}(w_\lambda) \leq \|\lambda_{min}\| \leq f(cost_{\mathcal{C}_\Gamma}(w_\lambda))$. By Proposition 3, item (2), Γ is bounded iff $\|\lambda_{min}\|$ is bounded over all λ. The latter condition holds iff $\mathcal{C}_\Gamma$ is limited over words w_λ, for all expansion λ. By definition, the latter is equivalent to $\mathcal{D}_\Gamma$ being limited. Summing up, we obtain that Γ is bounded iff $\mathcal{D}_\Gamma$ is limited, as required. Note that the whole construction can be done in exponential time. ◀

As a corollary to Proposition 12 and Theorem 8 we obtain the desired upper bound for part (1) of Theorem 11.

▶ **Corollary 13.** Boundedness *for UC2RPQs is in* ExpSpace.

Size of equivalent UCQs. Here we prove part (2) of Theorem 11. Since Γ is bounded we have from Proposition 12 that $\mathcal{D}_\Gamma$ is limited. Then, from Theorem 8 we obtain that the maximum cost that it takes $\mathcal{D}_\Gamma$ over a word is N, where N is exponential in the number of states of $\mathcal{D}_\Gamma$, and thus double exponential in $\|\Gamma\|$ by construction. Therefore, for every expansion λ of Γ, if λ_{min} is a minimal size expansion Γ such that $\lambda_{min} \to \lambda$, then $\|\lambda_{min}\| \leq f(N)$, where f is the polynomial function of the proof of Proposition 12. In particular, all minimal expansions of Γ are of size $\leq f(N)$. By Lemma 2, the UC2RPQ Γ is equivalent to the union of all its minimal expansions. The number of such minimal expansions is thus at most exponential in $f(N)$, and hence triple exponential in $\|\Gamma\|$.

6.2 Lower bounds

We reduce from the 2^n-tiling problem, that is, a tiling problem restricted to 2^n many columns, which is ExpSpace-complete (see, *e.g.*, [14]). We show that for every 2^n-tiling problem T there is a CRPQ γ, computable in polynomial time from T, whose number of minimal expansions is essentially the number of solutions to T in the following sense.

▶ **Lemma 14.** *For every 2^n-tiling problem T with m solutions there is a Boolean CRPQ γ, computable in polynomial time from T, such that the number of minimal expansions of γ is $O((g(|T|)+m)^{n+1})$ and $\Omega(m)$, for some double exponential function g. Further, γ consists of a Boolean CRPQ of the form $\exists x, y \bigwedge_{0\leq i\leq n}(x \xrightarrow{L_i} y)$, where each L_i is given as a regular expression.*

As a corollary, this yields an ExpSpace lower bound for the boundedness problem (part (1) of Theorem 11), as well as a triple exponential lower bound for the size of the UCQ equivalent to any bounded CRPQ (part (3) of Theorem 11), since one can produce 2^n-tiling problems having triple-exponentially many solutions.

7 Better-behaved Classes of UC2RPQs

Here we present two restrictions of UC2RPQs that exhibit a better behavior in terms of the complexity of Boundedness than the general case, namely, *acyclic UC2RPQs of bounded*

thickness and *strongly connected UCRPQs*. The improved bounds are PSPACE and Π_2^P, respectively, which turn out to be optimal.

Acyclic UC2RPQs of Bounded Thickness. For any two distinct variables x, y of a C2RPQ γ, we denote by $\text{Atoms}_\gamma(x, y)$ the set of atoms in γ of the form $x \xrightarrow{L} y$ or $y \xrightarrow{L} x$. The *thickness* of a C2RPQ γ is the maximum cardinality of a set of the form $\text{Atoms}_\gamma(x, y)$, for x, y variables of γ with $x \neq y$. The thickness of a UC2RPQ Γ is the maximum thickness over all the C2RPQs in Γ. The *underlying undirected graph* of γ has as vertex set the set of variables of γ and contains an edge $\{x, y\}$ iff $x \neq y$ and $\text{Atoms}_\gamma(x, y) \neq \emptyset$. A C2RPQ γ is *acyclic* if its underlying undirected graph is an acyclic graph (*i.e.*, a forest). A UC2RPQ Γ is acyclic if each C2RPQ in Γ is.

We show next that BOUNDEDNESS for acyclic UC2RPQs of bounded thickness is PSPACE-complete. These classes of UC2RPQs have been previously studied in the literature [4, 5]. In particular, it follows from [5, Theorem 4.2] that the containment problem for the acyclic UC2RPQs of bounded thickness is PSPACE-complete, and hence Theorem 15 below shows that BOUNDEDNESS is not more costly than containment for these classes.

▶ **Theorem 15.** *Fix $k \geq 1$. The problem* BOUNDEDNESS *is* PSPACE*-complete for acyclic UC2RPQs of thickness at most k.*

Proof (sketch). The lower bound follows directly from PSPACE-hardness of BOUNDEDNESS for RPQs (see Corollary 7). For the PSPACE upper bound, we follow a similar strategy as in the case of arbitrary UC2RPQs (Section 6.1), *i.e.*, we reduce boundedness of Γ to DA limitedness. The main difference is that, since Γ is acyclic, we can exploit the power of alternation and construct an A2DA$^\varepsilon$ $\mathcal{B}$ (instead of a 2DA, as in the proof of Proposition 12), such that Γ is bounded iff $\mathcal{B}$ is limited. The constant upper bound on the thickness of Γ implies that $\mathcal{B}$ is actually of polynomial size. The result follows then as limitedness of an A2DA$^\varepsilon$ can be decided in PSPACE in virtue of Theorem 10. ◀

Both conditions in Theorem 15, *i.e.*, acyclicity and bounded thickness, are necessary. Indeed, it follows from Lemma 14 that BOUNDEDNESS is EXPSPACE-hard even for:

- Boolean acyclic CRPQs.
- Boolean CRPQs of thickness one, whose underlying undirected graph is of *treewidth* two. Recall that the treewidth is a measure of how much a graph resembles a tree (cf., [20]) —acyclic graphs are precisely the graphs of treewidth one.

Indeed, the CRPQs of the form $\exists x, y \bigwedge_i (x \xrightarrow{L_i} y)$ used in Lemma 14 are Boolean and acyclic (but have unbounded thickness). Replacing each $(x \xrightarrow{L_i} y)$ with $(x \xrightarrow{\varepsilon} z_i) \wedge (z_i \xrightarrow{L_i} y)$, yields an equivalent CRPQ of thickness one whose underlying undirected graph has treewidth two.

Strongly Connected UCRPQs. We conclude this section with an even better behaved class of CRPQs in terms of BOUNDEDNESS. Unlike the previous case, the definition of this class depends on the underlying *directed* graph of a CRPQ γ. This contains a directed edge from variable x to y iff there is an atom in γ of the form $x \xrightarrow{L} y$. A CRPQ γ is *strongly connected* if its underlying directed graph is strongly connected, *i.e.*, every pair of variables is connected by some directed path. A UCRPQ Γ is strongly connected if every CRPQ in Γ is. We can then establish the following.

▶ **Theorem 16.** BOUNDEDNESS *is* Π_2^P*-complete for strongly connected UCRPQs.*

8 Discussion and Future Work

The main conclusion of our work is that techniques previously used in the study of containment of UC2RPQs can be naturally leveraged to pinpoint the complexity of Boundedness by using DA instead of NFA. This, however, requires extending results on limitedness to alternating and two-way DA. For all the classes of UC2RPQs studied in the paper we show in fact that the complexity of Boundedness coincides with that of the containment problem. We leave open what is the exact size of UCQ rewritings for the classes of acyclic UC2RPQs of bounded thickness and the strongly connected UCRPQs that are bounded.

The most natural next step is to study Boundedness for the class of *regular queries* (RQs), which are the closure of UC2RPQs under binary transitive closure. RQs are one of the most powerful recursive languages for which containment is decidable in elementary time. In fact, containment of RQs has been proved to be 2EXPSPACE-complete by applying sophisticated techniques based on NFA [34]. We will study if it is possible to settle the complexity of Boundedness for RQs with the help of DA techniques.

Another interesting future line of work is the study of Boundedness for UC2RPQs based on the restricted classes of regular expressions often found in practical applications [13]. As it has been shown lately, the complexity of some query evaluation problems is alleviated under this restriction [31], and it would be nice to see if the same holds for the boundedness problem. This would be good news for the applicability of boundedness techniques in practical applications. In fact, it would be an indication that the high complexity lower bounds obtained in this paper are mostly witnessed by complicated interactions between regular expressions not commonly arising in practice.

References

1 Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan L. Reutter, and Domagoj Vrgoč. Foundations of modern query languages for graph databases. *ACM Computing Surveys*, 50(5):68:1–68:40, 2017.

2 Pablo Barceló. Querying graph databases. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 175–188, 2013.

3 Pablo Barceló, Gerald Berger, Carsten Lutz, and Andreas Pieris. First-order rewritability of frontier-guarded ontology-mediated queries. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1707–1713, 2018.

4 Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Does query evaluation tractability help query containment? In *ACM Symposium on Principles of Database Systems (PODS)*, pages 188–199, 2014.

5 Pablo Barceló, Miguel Romero, and Moshe Y. Vardi. Semantic acyclicity on graph databases. *SIAM Journal on computing*, 45(4):1339–1376, 2016.

6 Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. A step up in expressiveness of decidable fixpoint logics. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 817–826, 2016.

7 Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 293–304. IEEE Computer Society Press, 2015. doi:10.1109/LICS.2015.36.

8 Meghyn Bienvenu, Peter Hansen, Carsten Lutz, and Frank Wolter. First order-rewritability and containment of conjunctive queries in horn description logics. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 965–971, 2016.

9 Jean-Camille Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical systems theory*, 26(3):237–269, 1993.

10 Achim Blumensath, Thomas Colcombet, Denis Kuperberg, Pawel Parys, and Michael Vanden Boom. Two-way cost automata and cost logics over infinite trees. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14*, pages 16:1–16:9. ACM Press, 2014. doi:10.1145/2603088.2603104.

11 Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Logical Methods in Computer Science (LMCS)*, 10(3), 2014.

12 Mikołaj Bojańczyk and Szymon Toruńczyk. Deterministic automata and extensions of weak MSO. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS)*, volume 4 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73–84. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2308.

13 Angela Bonifati, Wim Martens, and Thomas Timm. An analytical study of large SPARQL query logs. *PVLDB*, 11(2):149–161, 2017.

14 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Principles of Knowledge Representation and Reasoning (KR)*, pages 176–185, 2000.

15 Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Rewriting of regular expressions and regular path queries. *Journal of Computer and System Sciences (JCSS)*, 64(3):443–465, 2002.

16 Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Symposium on Theory of Computing (STOC)*, pages 77–90, 1977.

17 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5556 of *Lecture Notes in Computer Science*, pages 139–150. Springer, 2009. doi:10.1007/978-3-642-02930-1_12.

18 Thomas Colcombet and Christof Löding. The nesting-depth of disjunctive μ-calculus for tree languages and the limitedness problem. In *EACSL Annual Conference on Computer Science Logic (CSL)*, pages 416–430, 2008.

19 Stavros S. Cosmadakis, Haim Gaifman, Paris C. Kanellakis, and Moshe Y. Vardi. Decidable optimization problems for database logic programs (preliminary report). In *Symposium on Theory of Computing (STOC)*, pages 477–490, 1988.

20 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

21 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.

22 Haim Gaifman, Harry G. Mairson, Yehoshua Sagiv, and Moshe Y. Vardi. Undecidable optimization problems for database logic programs. *J. ACM*, 40(3):683–713, 1993.

23 Peter Hansen, Carsten Lutz, Inanç Seylan, and Frank Wolter. Efficient query rewriting in the description logic EL and beyond. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3034–3040, 2015.

24 Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *Journal of Computer and System Sciences (JCSS)*, 24(2):233–244, 1982.

25 Gerd G. Hillebrand, Paris C. Kanellakis, Harry G. Mairson, and Moshe Y. Vardi. Tools for datalog boundedness. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 1–12, 1991.

26 Daniel Kirsten. Distance desert automata and the star height problem. *Informatique Théorique et Applications (ITA)*, 39(3):455–509, 2005.

27 Dexter Kozen. Lower bounds for natural proof systems. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 254–266, 1977.

28 Denis Kuperberg and Michael Vanden Boom. Quasi-weak cost automata: A new variant of weakness. In *IARCS Annual Conference on Foundations of Software Technology and*

Theoretical Computer Science (FST&TCS), volume 13 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 66–77. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2011. doi:10.4230/LIPIcs.FSTTCS.2011.66.

29 Hing Leung. Limitedness theorem on finite automata with distance functions: An algebraic proof. *Theoretical Computer Science (TCS)*, 81(1):137–145, 1991. doi:10.1016/0304-3975(91)90321-R.

30 Hing Leung and Viktor Podolskiy. The limitedness problem on distance automata: Hashiguchi's method revisited. *Theoretical Computer Science (TCS)*, 310(1-3):147–158, 2004. doi:10.1016/S0304-3975(03)00377-3.

31 Wim Martens and Tina Trautner. Evaluation and enumeration problems for regular path queries. In *International Conference on Database Theory (ICDT)*, pages 19:1–19:21, 2018.

32 Jeffrey F. Naughton. Data independent recursion in deductive databases. *Journal of Computer and System Sciences (JCSS)*, 38(2):259–289, 1989.

33 Nir Piterman and Moshe Y. Vardi. From bidirectionality to alternation. *Theoretical Computer Science (TCS)*, 295:295–321, 2003. doi:10.1016/S0304-3975(02)00410-3.

34 Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular queries on graph databases. *Theoretical Computer Science (TCS)*, 61(1):31–83, 2017.

35 John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

36 Michael Vanden Boom. Weak cost monadic logic over infinite trees. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 580–591, 2011.

37 Michael Vanden Boom. *Weak cost automata over infinite trees*. PhD thesis, University of Oxford, UK, 2012.

A Appendix to Section 3

Proof of Lemma 2. The result is a straightforward consequence of the following known result.

▶ **Lemma 17.** [14] *Let Γ, Γ' be UC2RPQs. It is the case that $\Gamma \subseteq \Gamma'$ iff for each expansion λ of Γ there exists an expansion λ' of Γ' such that $\lambda \subseteq \lambda'$, or, equivalently, $\lambda' \to \lambda$.*

◀

Proof of Lemma 3. For (1)⇒(2), suppose that Γ is equivalent to a UCQ $\Phi = \bigvee_{1 \leq i \leq n} \varphi_i$. By Lemma 17, for every $1 \leq i \leq n$, there is an expansion λ_i of Γ with $\lambda_i \to \varphi_i$. We claim that (2) holds for $k = \max\{\|\lambda_i\| : 1 \leq i \leq n\}$. Let λ be an expansion of Γ. By Lemma 17, we have $\varphi_i \to \lambda$, for some $1 \leq i \leq n$, and then $\lambda_i \to \lambda$. Since $\|\lambda_i\| \leq k$, we are done. For the implication (2)⇒(3), note that if (2) holds for some $k \geq 1$, then the size of any minimal expansion of Γ is at most k. Finally, (3)⇒(1) follows directly from Lemma 2. ◀

B Appendix to Section 4

Proof of Proposition 5 . Note that for an RPQ or an existentially quantified RPQ γ whose input regular language is L, there is a bijection from L to the expansions of γ. For an RPQ L every expansion is minimal. In the case of a CRPQ $\gamma(x) = \exists y(x \xrightarrow{L} y)$ [resp. $\gamma(y) = \exists x(x \xrightarrow{L} y)$], where $x \neq y$, there is a bijection from L_{pf} [resp. L_{sf}] to the minimal expansions of γ. Finally, for a Boolean CRPQ $\gamma = \exists x, y(x \xrightarrow{L} y)$, with $x \neq y$, we have a bijection from L_{ff} to the minimal expansions. Then, the proposition follows directly from Proposition 3, item (3). ◀

Proof of Theorem 6. We focus on the lower bounds. We reduce from the following well-known PSPACE-complete problem: given a non-deterministic Turing machine M and a natural number n (given in unary) check whether M accepts the empty tape using n space. As usual, we encode configurations of M as words of length n over the alphabet $\mathbb{P} := \Sigma \cup (\Sigma \times S)$, where Σ and S are the tape alphabet and state set of M respectively. A run of M is then encoded by a word of the form $\#c_1 ¢ c_2 ¢ \cdots ¢ c_\ell \#$, where each c_i is an encoding of a configuration, $¢$ is used as a separator of configurations, and $\#$ to delimit the beginning and end of the run. We can assume without loss of generality that either M accepts the empty tape using n space and all non-deterministic branches in the computation of M accept before $|\mathbb{P}|^n$ steps; or any non-deterministic branch of M does not halt at all.

Given M and n as above, we can define an NFA of polynomial size that accepts the language $R := c_{\text{init}} \cdot (¢ \cdot C)^*$ over $\mathbb{P}$, where c_{init} encodes the (unique) initial configuration of M on the empty tape and C accepts all the words of length n over $\mathbb{P}$ that encode a configuration of M. We let T be the finite language that contains all the words of the form $c ¢ c'$, where c and c' encode configurations of M and c' *cannot* be reached from c in one step. Note that T can be accepted by a polynomial-sized NFA. We claim that M accepts the empty tape iff L_{pf} is finite for $L := \#R\# \cup \#(R¢ + \varepsilon)T$.

Assume first that M accepts the empty tape. Then every $w \in L_{\text{pf}}$ satisfies $|w| \leq 2 + (n+1)(|\mathbb{P}|^n + 1)$. Indeed, by contradiction, suppose that this is not the case for some $w \in L_{\text{pf}}$. If $w \in \#R\#$, then w cannot encode a run of M starting from the empty tape (as every such a run takes less than $|\mathbb{P}|^n$ steps). Then there exists $v \in \#(R¢ + \varepsilon)T$ with $|v| \leq 1 + (n+1)(|\mathbb{P}|^n + 1)$ that is a prefix of w. In particular, $|v| < |w|$ and then $w \notin L_{\text{pf}}$; a contradiction. Similarly, if $w \in \#(R¢ + \varepsilon)T$, then $w\#$ cannot encode a run of M starting

from the empty tape, and hence $w \notin L_{\text{pf}}$. Now suppose that M does not accept the empty tape. Then there are infinitely many words $w \in \#R\#$ encoding a run of M starting from the empty tape, which in particular belong to L_{pf}. Hence, L_{pf} is infinite.

Note that the same construction applies for the factor case, *i.e.*, M accepts the empty tape iff L_{ff} is finite for $L = \#R\# \cup \#(R\mathfrak{c} + \varepsilon)T$ (a simpler construction that still applies is $L = \#R\# \cup T$). Finally, we can reduce the prefix to the suffix case. Given an NFA accepting L, we construct an NFA accepting $L^R = \{w^R : w \in L\}$, where w^R is the *reverse* of w. Hence, L_{pf} is finite iff L^R_{sf} is finite. ◀

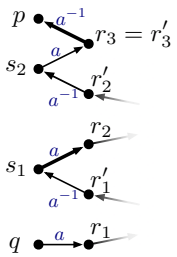
C Appendix to Section 5

Proof of Proposition 9. We adapt the standard "*crossing sequence*" construction for translating two-way NFA to one-way NFA [35]. For simplicity, and without any loss of generality, we assume that the input 2DA $\mathcal{A}$ is so that all accepting runs end with the head at the leftmost position, and that there is only one final state.

Given a 2DA $\mathcal{A} = (\mathbb{A}, Q, \emptyset, q_0, \{q_f\}, \delta)$ consider the DA $\mathcal{B} = (\mathbb{A}, Q', \emptyset, (q_0, q_f), F', \delta')$ where Q' is the set of subsets of $Q \times Q$, and F' is the set of subsets of $\{(q, q) : q \in Q\}$. The idea is that whenever a state contains a pair (q, p), it verifies that there is a loop at the current position in the run, starting in q and ending in p and visiting only positions which are to the right. Formally, there is a transition $(S, a, c, S') \in \delta'$ if for every pair $(q, p) \in S$ with $q \neq p$ there are $(r_1, r'_1), \ldots, (r_n, r'_n) \in S'$ so that

- there is a transition $(q, a, c', r_1) \in \delta$ and $(r'_n, a^{-1}, c', p) \in \delta$;
- for every r'_i with $i < n$ there are transitions $(r'_i, a^{-1}, c_i, s_i), (s_i, a, c_{i+1}, r_{i+1}) \in \delta$ for some $s_i \in Q$ and $c_i, c_{i+1} \in \{0, 1\}$;

and the maximum of the costs of the considered transitions is c. The following figure exemplifies the relation between (p, q) and the (r_i, r'_i)'s as seen in a run.



In this example, the loop (q, p) is witnessed, at the next position after reading an a, as the existence of two loops (r_1, r'_1) and (r_2, r'_2) (and the trivial loop (r_3, r'_3)).

From this construction it follows that $\mathcal{B}$ accepts the same language as $\mathcal{A}$, and that $cost_{\mathcal{B}}(w) \leq cost_{\mathcal{A}}(w)$ for every $w \in \mathbb{A}^*$. Further, note that without any loss of generality we can consider only states having no to distinct pairs with the same state, and thus states with at most $|Q|$ pairs. From this, it follows that $cost_{\mathcal{A}}(w) \leq cost_{\mathcal{B}}(w) \cdot 2 \cdot |Q|^2$. In the picture above, we are simulating 2 costly transitions (depicted with thick strokes) with only one costly transitions for the pair (q, p), and in general we could see $2|Q|$ transitions for each pair (q, p) in the state, hence $2|Q| \cdot |Q|$ costly transitions could be simulated at once. In fact, a finer analysis can show that $cost_{\mathcal{A}}(w) \leq cost_{\mathcal{B}}(w) \cdot 2 \cdot |Q|$. ◀

Proof of Theorem 10

Here we give the detailed proof of Theorem 10, namely, that the limitedness problem for A2DA is in PSpace.

(1) From A2DA$^\varepsilon$ to A2DA

We show a language-preserving polynomial reduction from A2DA$^\varepsilon$ to A2DA, obtained by replacing ε-transitions with a sequence of two transitions reading a and a^{-1} for some $a \in \mathbb{A}$. Formally, given an A2DA$^\varepsilon$ $\mathcal{A} = (\mathbb{A}, Q_\forall, Q_\exists, q_0, F, \delta)$, we produce an A2DA $\mathcal{B} = (\mathbb{A}, Q'_\forall, Q'_\exists, q_0, F', \delta')$ where $Q'_\forall$ contains $Q_\forall$ [resp. $Q'_\exists$ contains $Q_\exists$], plus fresh states $r_{q,p}$ and $r'_{q,p}$ for each $q \in Q_\forall$ [resp. $q \in Q_\exists$] and $p \in Q_\forall \cup Q_\exists$. The transition relation δ' is obtained from δ by replacing each transition $(q, \varepsilon, e, c, p)$ with all (polynomially many) transitions $(q, a, e', c, r_{q,p})$, $(q, a^{-1}, e, 0, r_{q,p})$, $(q, a^{-1}, e', c, r'_{q,p})$, $(r'_{q,p}, a, e, 0, p)$ for every $a \in \mathbb{A}$ and $e' \in \{0, 1\}$, where $r_{q,p}$ and $r'_{q,p}$ are fresh states of the same type as q (*i.e.*, $r_{q,p} \in Q'_\forall$ iff $q \in Q_\forall$ and likewise for $r'_{q,p}$). In other words, either we simulate the ε-transition by reading $a \cdot a^{-1}$ for some $a \in \mathbb{A}$ through $r_{q,p}$, or we simulate it by reading $a^{-1} \cdot a$ through $r'_{q,p}$. Note that if we would now define $F' = F$ we may not accept the empty word because we need words of length at least 1 to simulate ε-transitions. In order to fix this, we define $F' = F \cup \{q_0\}$ if some state of F can be reached from q_0 through a sequence of ε-transitions with *end* flag, or we define $F' = F$ otherwise.

The above reduction, although it does preserve the language, it does not preserve the cost of words: while the cost of the empty word can only be 0 for any A2DA, for an A2DA$^\varepsilon$ automaton it can be any arbitrary $n \in \mathbb{N}$. However, since it is a faithful simulation on non-empty words, for all $w \in \mathbb{A}^+$ we have $cost_\mathcal{A}(w) = cost_\mathcal{B}(w)$, and thus the reduction preserves the limitedness property.

(2) From A2DA to 2DA

Given an A2DA $\mathcal{A} = (\mathbb{A}, Q_\forall, Q_\exists, q_0, F, \delta)$, we build a 2DA $\mathcal{B}$ over a larger alphabet $\mathbb{B}$, where we trade alternation for extra alphabet letters. The alphabet $\mathbb{B}$ consists of triples $(f^\rightarrow, a, f^\leftarrow)$ where $a \in \mathbb{A}$ and $f^\rightarrow, f^\leftarrow : Q_\forall \to \delta$. The idea is that $f^\rightarrow, f^\leftarrow$ are "choice functions" for the alternation: whenever we are to the left [resp. right] of a position of the word labelled $(f^\rightarrow, a, f^\leftarrow)$ and we are in state $q \in Q_\forall$, instead of exploring all transitions departing from q and taking the maximum cost over all such runs (this is what alternation does), we chose to just take transition $f^\rightarrow(q)$ [resp. $f^\leftarrow(q)$]. Note that $\mathbb{B}$ is exponential in the number of states but not in the size of $\mathbb{A}$.

One can then build a 2DA $\mathcal{B}$ having the same set of states as $\mathcal{A}$ but with a transition function which is essentially deterministic on the states of $Q_\forall$, as it follows the choice function given by the alphabet letters. In the end we obtain that

- for every $w \in \mathbb{B}^*$, $cost_\mathcal{B}(w) \leq cost_\mathcal{A}(w_\mathbb{A})$; and
- for every $w \in \mathbb{A}^*$ there is $\widetilde{w} \in \mathbb{B}^*$ so that $\widetilde{w}_\mathbb{A} = w$ and $cost_\mathcal{A}(w) = cost_\mathcal{B}(\widetilde{w})$,[2]

where $w_\mathbb{A}$ is the projection of w onto $\mathbb{A}$. This shows that the limitedness problem is preserved.

For simplicity we assume that any state of $Q_\forall$ determines whether the head of the automaton moves rightwards or leftwards —it is easy to see that this is without loss of generality. Formally, we have that $Q_\forall$ is partitioned into two sets $Q_\forall = Q_\forall^\rightarrow \dot\cup Q_\forall^\leftarrow$ so that

[2] This can be alternatively seen as the existence of a positional strategy for the universal player for obtaining the cost $cost_\mathcal{A}(w)$ when the automata is seen as a two-player game.

there is no transition $(r, a, c, p) \in \delta^{\rightarrow}$ with $r \in Q_{\forall}^{\leftarrow}$, and no transition $(r, a, c, p) \in \delta^{\leftarrow}$ with $r \in Q_{\forall}^{\rightarrow}$, where $\delta^{\rightarrow}$ [resp. $\delta^{\leftarrow}$] is the set of all transitions from δ reading a letter from $\mathbb{A}$ [resp. from $\mathbb{A}^{-1}$].

More concretely, consider the alphabet $\mathbb{B} = \{f^{\rightarrow} : Q_{\forall}^{\rightarrow} \to \delta\} \times \mathbb{A} \times \{f^{\leftarrow} : Q_{\forall}^{\leftarrow} \to \delta\}$, and notice that

$$|\mathbb{B}| \leq |\mathbb{A}| \cdot |\delta|^{2 \cdot |Q_{\forall}|}. \tag{$\star$}$$

We now define a 2DA $\mathcal{B}$ so that $\mathcal{A}$ (over $\mathbb{A}$) is limited if, and only if, $\mathcal{B}$ (over $\mathbb{B}$) is limited. $\mathcal{B}$ has the same set of states as $\mathcal{A}$ but its transition function is essentially deterministic on the states of $Q_{\forall}$.

Concretely, let $\mathcal{B}$ be the 2DA defined as $(\mathbb{B}, \emptyset, Q_{\forall} \cup Q_{\exists}, q_0, F, \delta')$, where δ' is the union of

- $\{(r, (f^{\rightarrow}, a, f^{\leftarrow}), e, c, p) : r \in Q_{\exists} \wedge (r, a, e, c, p) \in \delta \wedge (f^{\rightarrow}, a, f^{\leftarrow}) \in \mathbb{B}\}$, that is, all transitions from δ starting from an existential state moving rightwards;
- $\{(r, (f^{\rightarrow}, a, f^{\leftarrow}), e, c, p) : r \in Q_{\forall}^{\rightarrow} \wedge f^{\rightarrow}(r) = (r, a, e, c, p) \wedge (f^{\rightarrow}, a, f^{\leftarrow}) \in \mathbb{B}\}$, that is, for any universal state, the transition defined by $f^{\rightarrow}$ moving rightwards if it can be applied;
- $\{(r, (f^{\rightarrow}, a, f^{\leftarrow}), e, c, p) : r \in Q_{\forall}^{\rightarrow} \wedge f^{\rightarrow}(r) = (r', a', e', c', p') \wedge (r, a, e) \neq (r', a', e') \wedge (r, a, e, c, p) \in \delta \wedge (f^{\rightarrow}, a, f^{\leftarrow}) \in \mathbb{B}\}$, that is, if for a universal state r, $f^{\rightarrow}(r)$ gives an inconsistent transition, disregard $f^{\rightarrow}$ and take any (consistent) transition from δ;

and similar sets for the case of the transitions moving leftwards:

- $\{(r, (f^{\rightarrow}, a, f^{\leftarrow})^{-1}, e, c, p) : r \in Q_{\exists} \wedge (r, a^{-1}, e, c, p) \in \delta \wedge (a, f^{\rightarrow}, f^{\leftarrow})^{-1} \in \mathbb{B}^{-1}\}$;
- $\{(r, (f^{\rightarrow}, a, f^{\leftarrow})^{-1}, e, c, p) : r \in Q_{\forall}^{\leftarrow} \wedge f^{\leftarrow}(r) = (r, a^{-1}, e, c, p) \wedge (f^{\rightarrow}, a, f^{\leftarrow})^{-1} \in \mathbb{B}^{-1}\}$;
- $\{(r, (f^{\rightarrow}, a, f^{\leftarrow})^{-1}, e, c, p) : r \in Q_{\forall}^{\leftarrow} \wedge f^{\leftarrow}(r) = (r', a', e', c', p') \wedge (r, a^{-1}, e) \neq (r', a', e') \wedge (r, a^{-1}, e, c, p) \in \delta \wedge (f^{\rightarrow}, a, f^{\leftarrow})^{-1} \in \mathbb{B}^{-1}\}$.

For any word $w \in \mathbb{B}^*$ we denote by $w_{\mathbb{A}} \in \mathbb{A}^*$ its projection onto $\mathbb{A}$.

▶ **Lemma 18.** *For every $w \in \mathbb{B}^*$, $cost_{\mathcal{B}}(w) \leq cost_{\mathcal{A}}(w_{\mathbb{A}})$.*

Proof. Let w be an arbitrary word over $\mathbb{B}$, $w = (f_1^{\rightarrow}, a_1, f_1^{\leftarrow}) \cdots (f_n^{\rightarrow}, a_n, f_n^{\leftarrow}) \in \mathbb{B}^*$. If there is no accepting run of $\mathcal{A}$ on $w_{\mathbb{A}}$, then there is no accepting run of $\mathcal{B}$ on w. Otherwise, suppose there is an accepting run t of $\mathcal{A}$ on $w_{\mathbb{A}} = a_1 \cdots a_n$ of cost N. We show that the functions $f_i^{\rightarrow}$'s and $f_i^{\leftarrow}$'s allow us to select a branch of t whose labelling $c_1 \cdots c_m \in \delta^*$ can be extended to an accepting run $c'_1 \cdots c'_m \in \delta'^*$ of $\mathcal{B}$ on w of cost $\leq N$. In this way, it follows that $cost_{\mathcal{B}}(w) \leq N \leq cost_{\mathcal{A}}(w_{\mathbb{A}})$.

Let us see how to obtain such a branch. Let us fix any homomorphism $h : t \to G_{\mathcal{A}, w_{\mathbb{A}}}$, given by the fact that t is an accepting run. Consider the following traversal of t, starting at the root. Whenever we are at a node x with $h(x) \in Q_{\exists} \times \{0, \ldots, |w|\}$, we go to the only child (unless it is the leaf, in which case the traversal ends); and whenever we are at a node x with $h(x) = (r, i) \in Q_{\forall}^{\rightarrow} \times \{0, \ldots, |w|\}$ [resp. $(r, i) \in Q_{\forall}^{\leftarrow} \times \{0, \ldots, |w|\}$], we go to the child obtained when taking the edge labelled $f_i^{\rightarrow}(r)$ [resp. $f_i^{\leftarrow}(r)$] if there is one, or to any child otherwise. Consider now the labeling $c_1 \cdots c_m \in \delta^*$ corresponding to the branch of t just described, and let $b_1 \cdots b_m \in (\mathbb{A} \cup \mathbb{A}^{-1})^*$ be the letters read by the transitions. For each $i \in \{1, \ldots, m\}$ with $b_i \in \mathbb{A}$ [resp. $b_i = a^{-1}$ with $a \in \mathbb{A}$], we define $c'_i \in \delta'$ as the result of replacing b_i with $(f_\ell^{\rightarrow}, b_i, f_\ell^{\leftarrow})$ [resp. replacing a^{-1} with $(f_\ell^{\rightarrow}, a, f_\ell^{\leftarrow})^{-1}$] in c_i, for $\ell = |b_1 \cdots b_i|$. It follows that $c'_1 \cdots c'_m$ is an accepting run of $\mathcal{B}$ on w with cost the same cost as $c_1 \cdots c_m$, which must be at most N. ◀

▶ **Lemma 19.** *For every $w \in \mathbb{A}^*$ there is $\widetilde{w} \in \mathbb{B}^*$ so that $\widetilde{w}_{\mathbb{A}} = w$ and $cost_{\mathcal{A}}(w) = cost_{\mathcal{B}}(\widetilde{w})$.*

Proof. Given a word $w \in \mathbb{A}^*$, we define a function $f_{\mathcal{A}, w} : \{0, \ldots, |w|\} \to Q_{\forall} \to \delta$ that maximizes the cost for w. That is, $f_{\mathcal{A}, w}(i)(q)$ is the transition that should be followed

whenever we are in state q at position i in order to obtain $cost_{\mathcal{A}}(w, q, i)$ —*i.e.*, to maximize the cost. (This function can be regarded as a positional strategy for the universal player obtaining the maximum cost in the two-player game associated with the A2DA automaton.) Formally, for any $q \in Q_\forall$, we define $f_{\mathcal{A},w}(i)(q)$ as any transition $t \in \delta$ so that $t \in \arg\max g_{i,q}$ for $g_{i,q} : \delta \to \mathbb{N} \cup \{\infty, -1\}$ defined as

$$g_{i,q}(r,a,c,p) = \begin{cases} c + cost_{\mathcal{A}}(w,p,i+1) & \text{if } i < |w|, a = w[i+1] \in \mathbb{A}, r = q; \\ c + cost_{\mathcal{A}}(w,p,i-1) & \text{if } i > 0, a = w[i]^{-1} \in \mathbb{A}^{-1}, r = q; \\ -1 & \text{otherwise.} \end{cases}$$

For any word $u \in \mathbb{A}^*$, let $\widetilde{u} \in \mathbb{B}^*$ be so that $|\widetilde{u}| = |u|$ and for each $1 \leq i \leq |u|$ we define $\widetilde{u}[i] = (f^{\rightarrow}, u[i], f^{\leftarrow})$, where $f^{\rightarrow}$ [resp. $f^{\leftarrow}$] is the restriction of $f_{\mathcal{A},u}(i-1)$ on the subdomain $Q_\forall^{\rightarrow}$ [resp. of $f_{\mathcal{A},u}(i)$ on $Q_\forall^{\leftarrow}$]. By definition of $f_{\mathcal{A},u}$ and $\mathcal{B}$, it follows that $cost_{\mathcal{A}}(w) = cost_{\mathcal{B}}(\widetilde{w})$. ◀

▶ **Lemma 20.** *$\mathcal{A}$ is limited if, and only if, $\mathcal{B}$ is limited.*

Proof. If $\mathcal{B}$ is limited, there is some $N \in \mathbb{N}$ so that $cost_{\mathcal{B}}(w) \leq N$ for every $w \in \mathbb{B}^*$. By Lemma 19, for every $w \in \mathbb{A}^*$ there is some $\widetilde{w} \in \mathbb{B}^*$ so that $cost_{\mathcal{A}}(w) = cost_{\mathcal{B}}(\widetilde{w}) \leq N$ and thus $\mathcal{A}$ is limited by N.

If $\mathcal{A}$ is limited, then there is some $N \in \mathbb{N}$ so that $cost_{\mathcal{A}}(w) \leq N$ for every $w \in \mathbb{A}^*$. Then, for every word $u \in \mathbb{B}^*$ we have $cost_{\mathcal{B}}(u) \leq cost_{\mathcal{A}}(u_{\mathbb{A}}) \leq N$ by Lemma 18. ◀

(3) From 2DA to ADA$^\varepsilon$

We show a polynomial-time translation from 2DA to ADA extended with ε-transitions, which preserves limitedness. In the case of finite automata, there exist language-preserving reductions from 2-way NFA to alternating 1-way NFA with a quadratic blowup [9, 33]. However, these translations, when applied blindly to reduce from 2DA to ADA, do not preserve the cost semantics nor limitedness of languages. On the other hand, [10] shows an involved construction that results in a reduction from 2DA to ADA on *infinite trees* (a more general and challenging setup), which preserves limitedness but it is not polynomial in the number of states. Here we give a self-contained translation from 2DA to ADA which serves our purpose: it preserves limitedness and it is polynomial time computable.

The translation is close to the language-preserving reduction from 2NFA to alternating 1NFA of [33], upgraded to take into account the cost of different alternation branches, somewhat in the same spirit as the *history summaries* from [10]. The reduction exploits the structure of the runs of 2-way finite automata, which can be described as "a tree of zig-zags", borrowing the wording of [33]. That is, every run of a two-way finite automaton on w can be seen as a tree of height at most $|w|$ whose nodes are labelled by $Q \cup Q^2$, in such a way that for each letter $a \in \mathbb{A}$, each pair of consecutive transitions reading a, a^{-1} induce a leaf and each pair of consecutive transitions reading a^{-1}, a induce a branching (*cf.* Figure 1). The idea is then to explore the tree top-down by spawning new threads at every branching and using only a statespace of $Q \cup Q^2$. Let us call a *zig-zag tree* to any such tree resulting from an accepting run of a 2DA.

Concretely, given a 2DA $\mathcal{A} = (\mathbb{A}, Q, \emptyset, q_0, F, \delta)$ we construct an ADA$^\varepsilon$

$$\mathcal{B} = (\mathbb{A}, Q_\exists \cup \{(q_0, end)\}, Q_\forall, (q_0, end), F', \delta')$$

where $Q_\exists = (Q \cup Q^2) \times \{0,1\} \times \{end, \overline{end}\}$ and $Q_\forall$, F' and δ' are of polynomial size. The idea is that during the run, a state $(q, p, c, e) \in Q \times Q \times \{0,1\} \times \{end, \overline{end}\}$ —which we

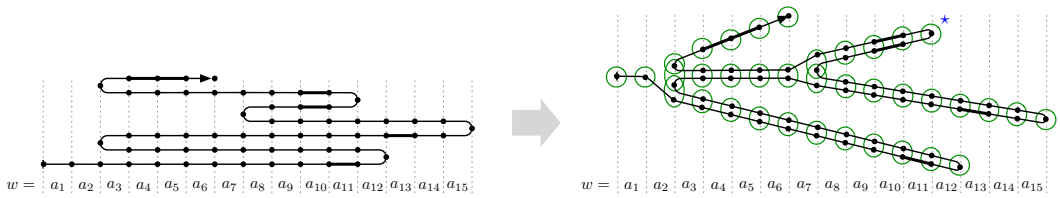


Figure 1 A run of 2DA seen as a tree of zig-zags. Tree nodes are depicted as circles, and there is an (implicit) edge from any circle to its right neighbor(s). Thick strokes represent costly transitions.

henceforth note as $[q,p]^e_c$— at position i verifies the presence of a 'right loop', that is, a partial run of $\mathcal{A}$ that starts in state q at i and ends in state p at i, visiting only positions $j \geq i$ to the right of i. The subscript c states whether the looping run contains at least one costly transition ($c = 1$) or no costly transitions ($c = 0$). The superscript e is simply a flag with the information of whether the current position is and end position or else, which we need for technical reasons. A state $\alpha \wedge \beta$ is understood as the alternation of states α and β. We will build an ADA with ε-transitions (*i.e.*, an ADA$^\varepsilon$), and for this reason we allow to have transitions $(q, a, e, 2, p)$ as short for $(q, a, e, 1, p'), (p', \varepsilon, e, 1, p)$ for a fresh state p'. Formally, $\mathcal{B}$ is defined as follows:

- $Q_\forall = \{\alpha \wedge \beta : \alpha, \beta \in Q_\exists\}$;
- $F' = \{[q]^e_0 : q \in F, e \in \{end, \overline{end}\}\} \cup \{[q,q]^e_0 : q \in Q, e \in \{end, \overline{end}\}\}$;
- δ' is defined as the smallest set verifying
  - $((q_0, end), \varepsilon, end, 0, [q_0]^{end}_c) \in \delta'$, for every $c \in \{0,1\}$;
  - $\{([q]^{e'}_1, a, e, 1, [p]^e_c) : c \in \{0,1\}, a \in \mathbb{A}, e, e' \in \{end, \overline{end}\}, (q, a, e, 1, p) \in \delta\} \subseteq \delta'$ —*i.e.*, every costly rightward transition of δ is in δ';
  - $\{([q]^{e'}_c, a, e, 0, [p]^e_c) : c \in \{0,1\}, a \in \mathbb{A}, e, e' \in \{end, \overline{end}\}, (q, a, e, 0, p) \in \delta\} \subseteq \delta'$ —*i.e.*, every non-costly rightward transition of δ is in δ';
  - for every $q, p \in Q$, $c_1, c_2 \in \{0,1\}$, and $e \in \{end, \overline{end}\}$ we have $([q]^e_{\max(c_1,c_2)}, \varepsilon, e, 0, [p]^e_{c_1} \wedge [q,p]^e_{c_2}) \in \delta'$ —*i.e.*, we can change the state from q to p provided there is a right loop from q to p;
  - for every $q, p, r \in Q$, $c_1, c_2 \in \{0,1\}$, and $e \in \{end, \overline{end}\}$, we have $([q,r]^e_{\max(c_1,c_2)}, \varepsilon, e, 0, [q,p]^e_{c_1} \wedge [p,r]^e_{c_2}) \in \delta'$ —*i.e.*, there is a right loop from q to r if there are from q to p and from p to r;
  - for every $a \in \mathbb{A}$, $q, p \in Q$, $c \in \{0,1\}$, $e_1, e_2 \in \{end, \overline{end}\}$ and $(q, a, e_1, c_1, q'), (p', a^{-1}, e_2, c_2, p) \in \delta$ we have
    - $([q,p]^{e_2}_c, a, e_1, 0, [q', p']^{e_1}_c) \in \delta'$ if $\max(c_1, c_2) = 0$,
    - $([q,p]^{e_2}_1, a, e_1, c_1 + c_2, [q', p']^{e_1}_c) \in \delta'$ if $\max(c_1, c_2) = 1$;

    —*i.e.*, a right loop on a position can be witnessed by a right loop on the next position;
  - for every $[\alpha_1]^e_{c_1} \wedge [\alpha_2]^e_{c_2} \in Q_\forall$ we have $([\alpha_1]^e_{c_1} \wedge [\alpha_2]^e_{c_2}, \varepsilon, e, c_2, [\alpha_1]^e_{c_1}) \in \delta'$ and $([\alpha_1]^e_{c_1} \wedge [\alpha_2]^e_{c_2}, \varepsilon, e, c_1, [\alpha_2]^e_{c_2}) \in \delta'$ —*i.e.*, $\alpha_1 \wedge \alpha_2$ is the alternation of states α_1 and α_2.

▶ **Lemma 21.** *$\mathcal{A}$ is limited if, and only if $\mathcal{B}$ is limited.*

Proof. First, note that the translation is language-preserving (*i.e.*, the set of words with accepting runs in $\mathcal{A}$ and in $\mathcal{B}$ coincide). Further, the accepting runs of $\mathcal{B}$ are essentially the accepting runs of $\mathcal{A}$ seen as zig-zag trees.

For any accepting run of $\mathcal{A}$ represented as a zig-zag tree, and any given branch thereof (*i.e.*, a path from the root to a leaf), let us define its *cost* as the number of costly transitions it contains. For example, in Figure 1 the branch indicated with $\star$ has cost 2. We also define the *number of heavy branchings* of a branch as the number of subtrees attached to the branch

that have at least one costly transition. In Figure 1 there are 3 subtrees attached to the $\star$-branch (which, in this particular case, they all look like words rather than trees), and all of them have costly transitions; hence the number of heavy branchings is 3. Finally, for any accepting run ρ of $\mathcal{A}$, let $f(\rho)$ be the maximum, over all its branches, of its cost plus its number of heavy branchings. Notice that $f(\rho) \leq \mathit{cost}(\rho)$. Observe also that for a word w, every accepting run t of $\mathcal{B}$ determines a zig-zag tree and then an accepting run ρ_t of $\mathcal{A}$. Conversely, for every accepting run ρ of $\mathcal{A}$, there is an accepting run t of $\mathcal{B}$ such that $\rho_t = \rho$. Further, the cost computed by $\mathcal{B}$ is closely related with f in the sense that $\frac{1}{k} f(\rho_t) \leq \mathit{cost}(t) \leq f(\rho_t)$ (†), for every accepting run t of $\mathcal{B}$, where $k := |Q|^2 + |Q|$ is the maximum arity of a zig-zag tree. Then we have the following:

$$\begin{aligned}\mathit{cost}_{\mathcal{B}}(w) &\leq \min\{f(\rho) : \rho \text{ is an accepting run of } \mathcal{A} \text{ on } w\} \\ &\leq \min\{\mathit{cost}(\rho) : \rho \text{ is an accepting run of } \mathcal{A} \text{ on } w\} = \mathit{cost}_{\mathcal{A}}(w),\end{aligned}$$

where $\min \emptyset = 0$. Therefore, we have that if $\mathcal{A}$ is limited, so is $\mathcal{B}$.

For the other direction, we claim that $\mathit{cost}(\rho) \leq 3k^{f(\rho)}$, for every accepting run ρ of $\mathcal{A}$. To see this, consider the heaviest branch B of ρ (*i.e.*, the result of traversing the tree from the root by always choosing a child whose subtree has maximal number of costly transitions). We can partition the edges of B into E_1 and E_2 such that E_1 are the edges that do not decrease the cost of the current subtree and E_2 the ones that do. Since the initial cost is $n = \mathit{cost}(\rho)$, and each edge in E_2 decreases the cost of the current subtree from n' to no less than $\frac{n'}{k} - 1$, we have $|E_2| \geq \max\{\ell \in \mathbb{N} : \frac{n}{k^\ell} - \sum_{i=0}^{\ell-1} \frac{1}{k^i} \geq 1\} \geq \max\{\ell \in \mathbb{N} : \frac{n}{k^\ell} - 2 \geq 1\} \geq \log_k n/3$. The claim follows since $f(\rho) \geq |E_2|$ (as each edge in E_2 is either costly or has a heavy branching).

From the bound above, we can obtain that $\mathit{cost}_{\mathcal{A}}(w) \leq 3k^{k \cdot \mathit{cost}_{\mathcal{B}}(w)}$, for every word w, and hence if $\mathcal{B}$ is limited, so is $\mathcal{A}$. Indeed, take an accepting run t of $\mathcal{B}$ with $\mathit{cost}(t) = \mathit{cost}_{\mathcal{B}}(w)$, and consider the associated accepting run ρ_t of $\mathcal{A}$. By (†), we have $f(\rho_t) \leq k \cdot \mathit{cost}(t)$. Summing up, we obtain $\mathit{cost}_{\mathcal{A}}(w) \leq \mathit{cost}(\rho_t) \leq 3k^{f(\rho_t)} \leq 3k^{k \cdot \mathit{cost}_{\mathcal{B}}(w)}$, as required. ◀

(4) From ADA$^\varepsilon$ to ADA

This is a straightforward polynomial time reduction. This reduction —as opposed to reduction (1)— does not preserve the language: we need to add an extra letter a_ε to the alphabet in order to make the reduction work in polynomial time. In fact, even for alternating finite automata (AFA) there is no known polynomial time language preserving translation from AFA with epsilon transitions into AFA (to the best of our knowledge). Given an ADA$^\varepsilon$ $\mathcal{A} = (\mathbb{A}, Q_\exists, Q_\forall, q_0, F, \delta)$, we can assume, without any loss of generality, that the state determines whether we are in a leftmost position, a rightmost position, or an internal position. That is, the statespace is partitioned into $Q_\exists = Q_{\exists,1} \dot\cup Q_{\exists,2} \dot\cup Q_{\exists,3}$ and $Q_\forall = Q_{\forall,1} \dot\cup Q_{\forall,2} \dot\cup Q_{\forall,3}$ so that $q_0 \in Q_{\exists,1}$ and every transition $(q, \alpha, e, c, p) \in \delta$ with $\alpha \in \mathbb{A} \cup \{\varepsilon\}$, $q \in Q_{\exists,i} \cup Q_{\forall,i}$ and $p \in Q_{\exists,j} \cup Q_{\forall,j}$ is so that: (i) $i \leq j$, (ii) $e = \mathit{end}$ iff $j \in \{1, 3\}$, (iii) if $\alpha = \varepsilon$ then $i = j$.

We obtain $\mathcal{B} = (\mathbb{B}, Q_\exists, Q_\forall, q_0, F, \delta')$ by extending the alphabet $\mathbb{A}$ with a new letter $\mathbb{B} = \mathbb{A} \cup \{a_\varepsilon\}$, and obtaining the ε-free transition relation δ' from δ by

- replacing each transition $(q, \varepsilon, e, c, p)$ with $(q, a_\varepsilon, e, c, p)$, and
- adding self-loops $(q, a_\varepsilon, \mathit{end}, 0, q)$ for each state $q \in \bigcup_{\dagger \in \{\exists,\forall\}, i \in \{1,3\}} Q_{\dagger,i}$ and $(q, a_\varepsilon, \overline{\mathit{end}}, 0, q)$ for each state $q \in Q_{\exists,2} \cup Q_{\forall,2}$.

It is easy to see that this reduction preserves limitedness.

(5) From ADA to DA

Finally, the last reduction is exactly the same as the reduction A2DA to 2DA, observing that when applied to an ADA it yields a DA. It is worth noting that a limitedness preserving reduction in the context of *infinite words* has been proposed in [28, Lemma 6], but it produces an automaton with an exponential set of states.

The resulting composition (1) + (2) + (3) + (4) + (5)

Let $\mathcal{A}_i = (\mathbb{A}_i, Q_{i,\exists}, Q_{i,\forall}, q_{i,0}, F_i, \delta_i)$ be an A2DA, for each $i \in \{1,2,3,4\}$ so that, starting with $\mathcal{A}_1$, we get $\mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ as the result from the reductions $\mathcal{A}_1 \xrightarrow{(1)+(2)} \mathcal{A}_2 \xrightarrow{(3)+(4)} \mathcal{A}_3 \xrightarrow{(5)} \mathcal{A}_4$ described before. We obtain the following properties.

- $\mathcal{A}_4$ has a polynomial number of states. More precisely, $|Q_{4,\forall}| = |Q_{2,\forall}| = 0$, and $|Q_{4,\exists}| = |Q_{3,\exists} \cup Q_{3,\forall}| \leq \mathit{poly}(|Q_{2,\exists}|) = \mathit{poly}(|Q_{1,\exists} \cup Q_{1,\forall}|)$.
- $\mathbb{A}_4$ is (single) exponential. Due to $(\star)$, $|\mathbb{A}_2| = |\mathbb{A}_1| \cdot |\delta_1|^{2 \cdot |Q_{1,\forall}|}$, $|\mathbb{A}_3| = |\mathbb{A}_2| + 1$, and $|\mathbb{A}_4| = |\mathbb{A}_3| \cdot |\delta_3|^{2 \cdot |Q_{3,\exists} \cup Q_{3,\forall}|}$ again due to $(\star)$. Since δ_3 is single exponential in $\mathcal{A}_1$ (since it is polynomial in $\mathbb{A}_3$ and $Q_{3,\exists}$), and $Q_{3,\exists}$ is polynomial in $\mathcal{A}_1$, $\mathbb{A}$ is single exponential in $\mathcal{A}_1$.

As explained before, thanks to the fact that the bound of Theorem 8 depends only on the number of states and not on the size of the alphabet nor the transition set, this enables a PSpace procedure for testing for limitedness of A2DA, which concludes the proof of Theorem 10.

D Appendix to Section 6

We encode an instance of the *tiling problem* following the ideas used for showing ExpSpace-hardness for CRPQ-containment [14]. We reduce from the following 2^n-tiling problem, which is ExpSpace-complete. An input instance consists of a number $n \in \mathbb{N}$ written in unary, a finite set Δ of tiles, two relations $H, V \subseteq \Delta \times \Delta$ specifying constraints on how tiles should be placed horizontally and vertically, and the starting and final tiles $t_S, t_F \in \Delta$. A solution to the input instance is a "consistent" assignment of tiles to a finite rectangle having 2^n columns. Concretely, a solution is a function $f : \{1, \ldots, 2^n\} \times \{1, \ldots, k\} \to \Delta$, for some $k \in \mathbb{N}$, such that $f(1,1) = t_S$, $f(2^n, k) = t_F$, and $f((i,j), f(i+1,j)) \in H$ and $f((i,j), f(i,j+1)) \in V$ for every i, j in range. We can then obtain the following.

▶ **Lemma** (restatement of Lemma 14). *For every 2^n-tiling problem T with m solutions there is a Boolean CRPQ γ, computable in polynomial time from T, such that the number of minimal expansions of γ is $O((g(|T|) + m)^{n+1})$ and $\Omega(m)$, for some double exponential function g. Further, γ consists of a Boolean CRPQ of the form $\exists x, y \bigwedge_{0 \leq i \leq n} (x \xrightarrow{L_i} y)$, where each L_i is given as a regular expression.*

Proof. For any tiling instance as above, we show how to define a CRPQ over the alphabet $\mathbb{A} := \Delta \cup \{0, 1, \#\}$ so that it has at least m and at most $(g(|T|) + m)^{n+1}$ minimal expansions for some double exponential function g, where m is the number of solutions of the instance. We will encode a solution of a tiling as a word of $\#((0+1)^n \cdot \Delta)^* \#$, where the rectangle of tiles is read left-to-right and top-to-bottom, and each block (*i.e.*, each element of $(0+1)^n\ \Delta$) represents the column number (in binary) and the tile. The symbols # at the beginning and end of the word are used for technical reasons.

For enforcing this encoding, we define regular languages E, F_C, F_H and G_i for each $i \leq n$ over $\mathbb{A}$.

The language E gives the general shape of the encoding of solutions:

$$E = \#0^n\ t_S\ ((0+1)^n\Delta)^*\ 1^n\ t_F\#,$$

in particular that it starts and ends with the correct tiles. The language F_C detects adjacent blocks with an error in the column number bit, which can be easily defined with a polynomial NFA. The language F_H checks that there are adjacent blocks in which the tiles do not respect the horizontal adjacency relation H,

$$F_H = \bigcup_{(t_1,t_2)\in\Delta^2\setminus H} t_1\ \overline{0^n}\ t_2,$$

where $\overline{0^n} = (0+1)^n \setminus \{0^n\}$. Finally, $G_0, \ldots, G_n$ are used to check that there are two blocks at distance 2^n which do not respect the vertical adjacency relation V; in other words, there is a factor of the word whose first and last blocks have the same column number, it contains not more than one block with column number 1^n (otherwise we would be skipping a row), and its first and last tiles are not V-related. First, G_0 checks that the first and last blocks of the factor we are interested in do not conform to V, and furthermore that there is exactly one column number 1^n in between

$$G_0 = \bigcup_{(t_1,t_2)\in\Delta^2\setminus V} (0+1)^n t_1(\overline{1^n}\ \Delta)^* 1^n(\Delta\cdot\overline{1^n})^* t_2,$$

where $\overline{1^n} = (0+1)^n \setminus \{1^n\}$. For each $b \in \{0,1\}$ and $i \in \{1,\ldots,n\}$ we define G_i^b to check that the i-th bit of the address of both the first and last tile is set to b,

$$G_i^b = (0+1)^{i-1}\cdot b\cdot(0+1)^{n-i}\cdot\Delta\cdot((0+1)^n\cdot\Delta)^*\cdot(0+1)^{i-1}b(0+1)^{n-i}\Delta,$$

and we define G_i as $G_i^0 + G_i^1$. For each one of these languages one can produce a regular expression recognizing the language in polynomial time. Finally, the Boolean CRPQ is

$$\gamma = \exists x, y \bigwedge_{0\le i\le n} x \xrightarrow{E\cup G_i\cup F_C\cup F_H} y.$$

Let us analyse the bounds on the number of minimal expansions of γ with respect to the number m of solutions of the tiling problem. First, note that for any expansion of γ containing a word $w \in E$ which *does not* encode a solution to the tiling problem either: (i) it has a problem with the column encodings, in which case it contains a (polynomial) word from F_C as a factor; (ii) the encoding is correct, but the horizontal relation is not respected, in which case it contains a (polynomial) word from F_H as a factor; (iii) it violates the vertical relation, and thus there are words $w_i \in G_i$ for each i so that $|w_i| \in O(n2^n)$ and the expansion corresponding to $w_0, \ldots, w_n$ maps to the expansion. Therefore, every path of a minimal expansion of γ which is in E and is not a solution cannot have size bigger than $O(n2^n)$, which means that the number of minimal expansions is at most $(|\mathbb{A}|^{O(n2^n)} + m)^{n+1}$. On the other hand, it follows by construction that every word encoding a solution is in E, that the expansion consisting of only solutions is minimal, and hence that there are at least m minimal expansions of γ. ◀

Without loss of generality we can assume that the tiling instance satisfies that if there is a tiling solution, there are infinitely many. This fact, coupled with Lemma 14 and ExpSpace-completeness of the 2^n-tiling problem, yields the lower bound in part (1) of Theorem 11.

▶ **Proposition 22.** Boundedness *for CRPQs is* ExpSpace*-hard. This holds even for Boolean CRPQs of the form* $\exists x, y \bigwedge_i (x \xrightarrow{L_i} y)$*, whose languages* L_i *are given as regular expressions.*

Lower bound on size of equivalent UCQ. It is not hard to produce 2^n-tiling instances T_n having triple-exponentially many solutions. Indeed, it suffices to (1) enforce that each solution has exactly 2^{2^n} rows (and hence there are only finitely many solutions), which can be done by encoding the binary representation of i at each row i, and (2) encode at each row an arbitrary symbol from the alphabet $\{a, b\}$. In this way, each solution encodes a function $f : R \to \{a, b\}$, where $R = \{0, \ldots, 2^{2^n} - 1\}$, and conversely, for each such a function there is a distinct solution. It then follows that T_n has $2^{2^{2^n}}$ solutions. In particular, the Boolean CRPQ γ_n from Lemma 14 is bounded and has at least $2^{2^{2^n}}$ minimal expansions. Recall that these minimal expansions are produced by expanding each atom of γ_n into a word $w \in E$ corresponding to a solution of T_n. Hence, if λ and λ' are two of these minimal expansions, we have that $\lambda \not\to \lambda'$, *i.e.*, γ_n has at least $2^{2^{2^n}}$ homomorphically incomparable minimal expansions. By Lemma 17, it follows that every UCQ equivalent to γ_n must have at least $2^{2^{2^n}}$ disjuncts. This yields part (3) of Theorem 11.

E Appendix to Section 7

Proof of Theorem 15

The PSpace lower bound follows from Corollary 7, so we focus on the upper bound. Given an acyclic UC2RPQ Γ of thickness $\leq k$, we shall construct in polynomial time an A2DA$^\varepsilon$ $\mathcal{A}$ of polynomial size in $\|\Gamma\|$ such that Γ is bounded iff $\mathcal{A}$ is limited. The result will follow from Theorem 10.

As in the proof of the ExpSpace upper bound in Theorem 11, the A2DA$^\varepsilon$ $\mathcal{A}$ will run over encodings of expansions of Γ. So if $\mathbb{A}$ is the alphabet of Γ, then the alphabet of $\mathcal{A}$ is $\mathbb{A}_1 := \mathbb{A}^\pm \cup \mathcal{V} \cup \{\$\}$, where $\mathcal{V}$ is the set of variables of Γ and \$ is a fresh symbol. Again, if λ is the expansion of a disjunct $\gamma = \exists \bar{z} \bigwedge_{1 \leq i \leq m} (x_i \xrightarrow{L_i} y_i)$ of Γ obtained by expanding each $x_i \xrightarrow{L_i} y_i$ into an oriented path π_i from x_i to y_i with label $w_i \in L_i$, then we encode λ as the word over $\mathbb{A}_1$

$$w_\lambda = \$x_1 w_1 y_1 \$ x_2 w_2 y_2 \$ \cdots \$ x_m w_m y_m \$$$

Note how the subword $x_i w_i y_i$ represents the oriented path π_i. Every position $j \in \{1, \ldots, |w_\lambda|\}$ with $w_\lambda[j] \neq \$$ represents a variable in λ: either x_i or y_i if $w_\lambda[j] = x_i$ or $w_\lambda[j] = y_i$, respectively; or the $(\ell + 1)$-th variable in the oriented path π_i if $w_\lambda[j]$ is the ℓ-th symbol in the subword w_i. Hence different positions could represent the same variable in λ: *e.g.*, in the encoding $\$xabcy\$$, the $5th$ position containing a 'c' and $6th$ position containing a 'y', represent the same variable, namely, the last vertex y of the oriented path.

It follows from the definition of $\mathcal{B}_2^\gamma$ and $\mathcal{D}_2^\gamma$ in Section 6.1 that for every regular language L appearing in Γ, there is a 2DA $\mathcal{A}_L$ over $\mathbb{A}_1$, computable in polynomial time, such that for every expansion λ of Γ, and head positions $i, j \in \{1, \ldots, |w_\lambda|\}$, where x_i and x_j are the variables in λ represented by i and j, respectively, we have:

- Every accepting run ρ of $\mathcal{A}_L$ over w_λ from position i to position j, determines an oriented path π_ρ in λ from x_i to x_j whose label is in L. Moreover, $cost(\rho)$ is precisely the number of atoms of π_ρ.
- For every oriented path π in λ from x_i to x_j with label in L, there is an accepting run ρ of $\mathcal{A}_L$ over w_λ from position i to position j, such that $\pi_\rho = \pi$.

Recall also from Section 6.1 that there is an NFA $\mathcal{A}_1$ that accepts precisely those words over $\mathbb{A}_1$ that encode some expansion of Γ. The size of $\mathcal{A}_1$ is polynomial in $\|\Gamma\|$. From $\mathcal{A}_1$ we can obtain a 2NFA $\mathcal{C}_1$ that, over a word w, starts by executing $\mathcal{A}_1$ until we reach the position $|w|$ and if we reach a final state of $\mathcal{A}_1$ then we move to the position 0 and accept, *i.e.*, we enter the final state of $\mathcal{C}_1$. Let $\mathcal{D}_1$ be the DA obtained from $\mathcal{C}_1$ by setting the cost of all transitions to be 0. Our A2DA$^\varepsilon$ $\mathcal{A}$ is the concatenation of $\mathcal{D}_1$ and $\mathcal{B}$ (defined below), *i.e.*, we add non-costly ε-transitions from the unique final state of $\mathcal{D}_1$ to the initial state of $\mathcal{B}$. Note that a word that is not of the form w_λ for some expansion λ of Γ has no accepting run of $\mathcal{A}$. Hence $\mathcal{A}$ is limited iff it is limited over the words w_λ's. Moreover, we have $cost_\mathcal{A}(w_\lambda) = cost_\mathcal{B}(w_\lambda)$. Thus, $\mathcal{A}$ is limited iff $\mathcal{B}$ is limited over all words of the form w_λ. We shall construct $\mathcal{B}$ so the latter condition is equivalent to Γ being bounded. In particular, for an expansion λ of Γ, let λ_{min} be a minimal size expansion of Γ such that $\lambda_{min} \to \lambda$. We will show that

$$cost_\mathcal{B}(w_\lambda) \leq \|\lambda_{min}\| \leq g(cost_\mathcal{B}(w_\lambda)), \text{ for every expansion } \lambda, \tag{1}$$

where g is some non-decreasing function. Hence, $\mathcal{B}$ is limited over all words of the form w_λ iff $\|\lambda_{min}\|$ is bounded over all expansions λ of Γ. The latter condition is equivalent to boundedness of Γ by our characterization of Proposition 3, item (2).

Before defining $\mathcal{B}$ we need to introduce some notation. Let γ be a disjunct of Γ. A *connected component* of γ is a maximal subquery whose underlying graph is connected. Since γ is acyclic, we can assume that (the underlying graph of) every connected component of γ is a rooted tree, and we use the usual terminology of trees (parent, children, leaves, ...) over the variables of γ. For variables x, y in γ, we define $\text{Atoms}_\gamma(x, y)$ to be the set of atoms of γ of the form $x \xrightarrow{L} y$ or $y \xrightarrow{L} x$. Suppose x is the parent of y in γ. Without loss of generality, we shall assume that each atom in $\text{Atoms}_\gamma(x, y)$ is of the form $x \xrightarrow{L} y$ (otherwise, we simply "reverse" L). We also assume a fixed enumeration $L_1, \ldots, L_\ell$, where $\ell = |\text{Atoms}_\gamma(x, y)|$, of the regular languages labelling the atoms of $\text{Atoms}_\gamma(x, y)$. We define $\text{Cuts}_\gamma(x, y)$ to be the set of *cuts* from x to y, that is, the set of tuples $(q_1, \ldots, q_\ell)$, where each q_i is a state of $\mathcal{A}_{L_i}$. We say that $(q_1, \ldots, q_\ell)$ is an *initial cut* if each q_i is the initial state of $\mathcal{A}_{L_i}$. Similarly, we say that $(q_1, \ldots, q_\ell)$ is a *final cut* if each q_i is a final state of $\mathcal{A}_{L_i}$. We also define $\text{Cuts}_\gamma = \bigcup\{\text{Cuts}_\gamma(x, y) : x \text{ parent of } y \text{ in } \gamma\}$ and $\text{Triples}_\gamma = \{(L, q, q') : L \text{ appears in } \gamma \text{ and } q, q' \text{ states in } \mathcal{A}_L\}$.

The main idea of the A2DA$^\varepsilon$ $\mathcal{B}$ is similar to the idea behind Section 6.1: for an encoding w_λ of an expansion λ of Γ, the automaton $\mathcal{B}$ tries to map some expansion λ' of Γ into (the encoding of) λ. In particular, every accepting run ρ of $\mathcal{B}$ over w_λ determines an expansion λ_ρ such that $\lambda_\rho \to \lambda$. On the other hand, for every expansion λ' such that $\lambda' \to \lambda$ there is an accepting run ρ of $\mathcal{B}$ over w_λ with $\lambda_\rho = \lambda'$. We will prove that

$$cost(\rho) \leq \|\lambda_\rho\| \leq g(cost(\rho)), \text{ for every accepting run } \rho \text{ of } \mathcal{B} \text{ over } w_\lambda, \tag{2}$$

where g is some non-decreasing function. Note then that (2) implies (1).

The main difference from the construction of Section 6.1 is that we do not need to annotate first the encoding w_λ and then project the annotations. Instead, we can exploit the acyclicity of Γ to try to map directly an expansion of Γ to the input expansion λ. To do this, $\mathcal{B}$ starts by choosing a disjunct γ of Γ. Then $\mathcal{B}$ applies universal transitions to map all the connected components of γ. Each component is mapped in a top-down fashion starting from the root to the leaves. Once some variable x is already mapped to some variable $h(x)$ of λ, or more precisely, to some head position $j_x \in \{0, \ldots, |w_\lambda|\}$ such that $j_x > 0$ and $w_\lambda[j_x]$ represents the variable $h(x)$, then $\mathcal{B}$ applies universal transitions to what we call the *axes* of

x. An axis of x is either an atom $x \xrightarrow{L} x \in \text{Atoms}_\gamma(x,x)$ or a child of x in γ. We need to extend our mapping to every axis of x.

By definition of the 2DA $\mathcal{A}_L$, the mappings of an axis $x \xrightarrow{L} x \in \text{Atoms}_\gamma(x,x)$ correspond to the accepting runs of $\mathcal{A}_L$ over w_λ starting and ending at position j_x. In order to find these looping accepting runs of $\mathcal{A}_L$, we use a similar idea as in the reduction (3) in Theorem 10 from 2DA to ADA$^\varepsilon$. We have that every accepting run of $\mathcal{A}_L$ over w_λ starting and ending at position j_x, can be divided into a set of rightward looping partial runs and a set of leftward looping partial runs. Following the terminology of reduction (3), we can represent rightward and leftward looping partial runs by *rightward* and *leftward zig-zag trees*, respectively, which are rooted trees of height at most $|w_\lambda|$ where each edge has a cost in $\{0,1,2\}$ and each node is labeled with (L,q,q') for a pair of states q,q' in $\mathcal{A}_L$. In a rightward [resp. leftward] zig-zag tree, each level ℓ from the root to the leaves corresponds to the position $j_x+\ell$ in $\{0,\ldots,|w_\lambda|\}$ [resp. $j_x-\ell$]; *cf.*, Figure 1. If u has only one child v in a rightward zig-zag tree (the leftward case is analogous) and their labels are (L,q,p) and (L,q',p'), respectively, then there must be transitions of the form (q,a,c_1,q') and (p',a^{-1},c_2,p) in $\mathcal{A}_L$ such that $a = w_\lambda[j_x+\ell_v]$, where ℓ_v is the level of v (*i.e.*, its distance to the root). The cost of the edge $\{u,v\}$ is then c_1+c_2. If u has children $v_1,\ldots,v_r$, with $r \geq 2$, and the label of u is (L,q,p) then the labels of $v_1,\ldots,v_r$ must be $(L,q_0,q_1),(L,q_1,q_2),\ldots,(L,q_{r-1},q_r)$, respectively, and there must be transitions (q,a,c_1,q_0) and (q_r,a^{-1},c_2,p) of $\mathcal{A}_L$ such that $a = w_\lambda[j_x+\ell]$, where ℓ is the level of the v_i's. The cost of $\{u,v_i\}$ is c_1 for $i=1$, c_2 for $i=r$, and 0 for $1<i<r$. Finally, every leaf in a rightward or leftward zig-zag tree has a label of the form (L,q,q), for some state q in $\mathcal{A}_L$.

In order to map $x \xrightarrow{L} x$, the automaton $\mathcal{B}$ chooses a final state q' of $\mathcal{A}_L$ and enters a state $(L,q,q')^\varepsilon$, where q is the initial state of $\mathcal{A}_L$. From there, $\mathcal{B}$ can spawn threads starting from states (L,q,p,s) and $(L,p,q')^\varepsilon$, where $s \in \{\text{right},\text{left}\}$ and the state (L,q,p,s) indicates that we are looking for a s-ward partial run ρ from q to p starting and ending at j_x. As in the reduction (3), this is done by exploiting alternation to guess the zig-zag tree of ρ in a top-down manner. In order to compute the cost of an accepting run correctly, as in reduction (3), the states of $\mathcal{B}$ of the form (L,q,p,s) are enhanced with a number $c \in \{0,1\}$ (and denoted by $[(L,q,p,s)]_c$), which indicates whether there is a costly transition in the looping partial run from q to p (or equivalently, in the zig-zag subtree rooted at (L,q,p)).

For an axis corresponding to a child y of x, we need to map simultaneously all the atoms $x \xrightarrow{L_1} y$, …, $x \xrightarrow{L_\ell} y \in \text{Atoms}_\gamma(x,y)$. The idea is first to choose whether y is mapped to the right or to the left of j_x, and then moving in the chosen direction guessing a sequence of cuts $D_0,\ldots,D_n \in \text{Cuts}_\gamma(x,y)$, where D_0 is an initial cut and D_n is final. These are represented by $\mathcal{B}$ via states (D_i,s), where $s \in \{\text{right},\text{left}\}$. A transition from D_i to D_{i+1} can either consume a symbol from w_λ or be an ε-transition that modifies only one coordinate of D_i, say j, and produces D_{i+1}. When the latter happen, $\mathcal{B}$ spawn threads starting in states (D_{i+1},s) and (L_j,q,q',s'), where q and q' are the j-th coordinates of D_i and D_{i+1}, respectively, and $s' \in \{\text{right},\text{left}\}$ is some direction. The intuition is that we choose to do an asynchronous mapping only for $\mathcal{A}_{L_j}$ in the form of a s'-ward looping partial run from q to q'. This is needed as the mappings of the atoms $x \xrightarrow{L_j} y$ into w_λ can be very different from each other.

Observe that we can represent mappings of $\{x \xrightarrow{L_1} y, \ldots, x \xrightarrow{L_\ell} y\} = \text{Atoms}_\gamma(x,y)$ into w_λ, where x and y are mapped to j_x and j_y (we assume $j_y \geq j_x$; the case $j_y \leq j_x$ is analogous), respectively, as *rightward special trees* from j_x to j_y (or leftward in the case $j_y \leq j_x$), which are rooted trees with costs in the edges obtained from a special rooted path B with $j_y - j_x + 1$ nodes, each of which is labeled with a pair of cuts (D,D') in $\text{Cuts}_\gamma(x,y)$,

by attaching some zig-zag trees to each node as explained below. If (D, D') is the label of the root, *i.e.*, the first node of B, then D is an initial cut, and if (D, D') is the label of the last node of B then D' is a final cut. If $\{u, v\}$ is the r-th edge of B with $r \in \{1, \ldots, j_y - j_x\}$, and (D, D') and (E, E') are the labels of u and v respectively, then for every coordinate $i \in \{1, \ldots, \ell\}$, there is a transition in $\mathcal{A}_{L_i}$ of the form (q, a, c_i, q'), where q and q' are the i-th coordinates of D' and E, respectively, and $a = w_\lambda[j_x + r]$. The cost of the edge $\{u, v\}$ is $c_1 + \cdots + c_\ell \in \{0, \ldots, \ell\}$. In a rightward special tree we also have that each node u in the special rooted path B is associated with a set of disjoint rightward or leftward zig-zag trees whose roots have been identified with u. In particular, if u has label (D, D') and q_i and q'_i are the i-th coordinates of D and D', respectively, then for every $i \in \{1, \ldots, \ell\}$ there is a sequence $p_i^0, p_i^1, \ldots, p_i^{n_i}$ such that $p_i^0 = q_i$, $p_i^{n_i} = q'_i$ and the set of labels of the roots of the rightward or leftward zig-zag tree associated with u (before being identified with u) is precisely $\{(L_i, p_i^j, p_i^{j+1}) : i \in \{1, \ldots, \ell\}, j \in \{0, \ldots, n_i - 1\}\}$. Observe that the working of $\mathcal{B}$ explained in the previous paragraph for mapping $x \xrightarrow{L_1} y, \ldots, x \xrightarrow{L_\ell} y \in \mathrm{Atoms}_\gamma(x, y)$ into w_λ can be seen as using the power of alternation to guess a rightward or leftward special tree from the current position j_x to some position j_y. Again, in order to compute the cost correctly, we need to consider states of the form $[(D, s)]_c$, for $c \in \{0, 1\}$, instead of (D, s).

Now we are ready to formally define the A2DA$^\varepsilon$ $\mathcal{B}$. We shall use transitions of the form (q, a, c, q') where $c \in \{0, \ldots, k\}$. Note that this is not a problem as they can be simulated using ε-transitions. Moreover, the automaton $\mathcal{B}$ will not need to move its head to the leftmost and rightmost positions (as these never represent a variable of the input expansion), so almost all of its transitions will be $\overline{end}$-flagged. Hence, for a transition, we will write (q, a, c, q') instead of $(q, a, \overline{end}, c, q')$. We define $\mathcal{B} = (\mathbb{A}_1, Q_\exists, Q_\forall, q_0, F, \delta)$, where

- $Q_\exists = \{q_0, q_f\} \cup \bigcup\{Q_\exists^\gamma : \gamma \text{ disjunct of } \Gamma\}$. For a disjunct γ of Γ, we define

$$Q_\exists^\gamma = ((\mathrm{Cuts}_\gamma \cup \mathrm{Triples}_\gamma) \times \{\text{right,left}\} \cup \mathrm{Triples}_\gamma^\varepsilon) \times \{0, 1\} \cup \mathrm{Axes}_\gamma \cup \mathrm{Roots}_\gamma,$$

  where $\mathrm{Triples}_\gamma^\varepsilon := \{t^\varepsilon : t \in \mathrm{Triples}_\gamma\}$,

$$\mathrm{Axes}_\gamma := \{x_y : x \text{ is the parent of } y \text{ in } \gamma\} \cup \{x_A : x \text{ is in } \gamma,\ A \in \mathrm{Atoms}_\gamma(x, x)\},$$

  and $\mathrm{Roots}_\gamma := \{r_{\mathrm{init}} : r \text{ is the root of some connected component of } \gamma\}$. We shall write $[\alpha]_c$ for $(\alpha, c) \in Q_\exists^\gamma \setminus (\mathrm{Axes}_\gamma \cup \mathrm{Roots}_\gamma)$.
- $Q_\forall = \bigcup\{\{q_0^\gamma\} \cup Q_\forall^\gamma : \gamma \text{ disjunct of } \Gamma\} \cup \{q_{\mathrm{nf}}\}$. For a disjunct γ of Γ, we define $Q_\forall^\gamma = \mathcal{V}_\gamma \cup \{[\alpha_1]_{c_1} \wedge [\alpha_2]_{c_2} : [\alpha_1]_{c_1} \in Q_\exists^\gamma \setminus (\mathrm{Axes}_\gamma \cup \mathrm{Roots}_\gamma), [\alpha_2]_{c_2} \in \mathrm{Triples}_\gamma \times \{\text{right}, \text{left}\} \times \{0, 1\}\}$.
- $F = \{q_f\} \cup \bigcup\{[(L, q, q, s)]_0, [(L, q, q)^\varepsilon]_0 : \gamma \text{ disjunct of } \Gamma, (L, q, q) \in \mathrm{Triples}_\gamma, s \in \{\text{right}, \text{left}\}\}$.
- δ is the smallest set verifying:

  1. $(q_0, \varepsilon, \mathit{end}, 0, q_0^\gamma) \in \delta$, for every disjunct γ of Γ, so we can choose the disjunct of Γ to be mapped into the input expansion.
  2. $(q_0^\gamma, \varepsilon, \mathit{end}, 0, r_{\mathrm{init}}) \in \delta$, for every disjunct γ of Γ and every root r of a connected component of γ. For the disjunct γ to be mapped, we need all of its connected components to be mapped.
  3. For every disjunct γ of Γ, every root r of a connected component of γ, and every $a \in \mathbb{A}_1^\pm$, we have $\{(r_{\mathrm{init}}, a, 0, r_{\mathrm{init}}), (r_{\mathrm{init}}, \varepsilon, 0, r)\} \subseteq \delta$. With these transitions we can choose the position where the root r should be mapped.
  4. $(x, \varepsilon, 0, x_t) \in \delta$, for every disjunct γ of Γ, every x in γ and every $x_t \in \mathrm{Axes}_\gamma$. Once we mapped x into the expansion, we need to map all of its subtrees and all atoms in $\mathrm{Atoms}_\gamma(x, x)$.

5. For every γ in Γ, and every free variable x in γ, we have $(x, x^{-1}, 0, q_f) \in \delta$ and $(x, b^{-1}, 0, q_{\text{nf}}) \in \delta$, for every $b \in \mathbb{A}_1 \setminus \{x\}$. This ensures that x is always mapped to itself.
6. For every γ in Γ and $x_t \in \text{Axes}_\gamma$,
   - If $t = y$ (and hence x is the parent of y in γ), then $(x_t, \varepsilon, 0, [(D, s)]_c) \in \delta$, for every $c \in \{0, 1\}$, c and every initial cut $D \in \text{Cuts}_\gamma(x, y)$.
   - If $t = x \xrightarrow{L} x$, then $(x_t, \varepsilon, 0, [(L, q, q')^\varepsilon]_c) \in \delta$, where q is the initial state of $\mathcal{A}_L$, every final state q' of $\mathcal{A}_L$, and every $c \in \{0, 1\}$.

   These transitions allow us to start looking for a mapping of each axis of x into the expansion.
7. For every γ in Γ,
   - $([(D, s)]_{\max(c_1,c_2)}, \varepsilon, 0, [(D', s)]_{c_1} \wedge [(L, q, q', s')]_{c_2}) \in \delta$, for every $c_1, c_2 \in \{0, 1\}$, $s, s' \in \{\text{right}, \text{left}\}$ and every $D, D' \in \text{Cuts}_\gamma(x, y)$, for some x, y, such that $D = (q_1, \ldots, q_\ell)$, $D' = (q'_1, \ldots, q'_\ell)$, there is j such that $q_i = q'_i$, for all $i \in \{1, \ldots, \ell\} \setminus \{j\}$, and $(L, q, q') = (L_j, q_j, q'_j)$, where L_j is the j-th language mentioned in $\text{Atoms}_\gamma(x, y)$. With these transitions we guess that a looping s'-ward partial run of $\mathcal{A}_L$ from q to q' should be mapped to the input expansion. In terms of special trees, these transitions allow us to add new subtrees to a node in the special path of the s-ward special tree.
   - $([(L, q, q')^\varepsilon]_{\max(c_1,c_2)}, \varepsilon, 0, [(L, p, q')^\varepsilon]_{c_1} \wedge [(L, q, p, s)]_{c_2}) \in \delta$, for every $c_1, c_2 \in \{0, 1\}$, $s \in \{\text{right}, \text{left}\}$ and q, q', p states in $\mathcal{A}_L$. With these transitions we can guess that a looping s-ward partial run of $\mathcal{A}_L$ from q to p should be mapped to the input.
   - $([(L, q, q', s)]_{\max(c_1,c_2)}, \varepsilon, 0, [(L, q, p, s)]_{c_1} \wedge [(L, p, q', s)]_{c_2}) \in \delta$, for every $c_1, c_2 \in \{0, 1\}$, $s \in \{\text{right}, \text{left}\}$ and q, q', p states in $\mathcal{A}_L$. We reduce the search for the looping s-ward partial run of $\mathcal{A}_L$ from q to q', to look for loopings s-ward partial runs from q to p and from p to q'. In terms of zig-zag trees, these transitions allow us to add a new subtree to the s-ward zig zag tree.
8. For every γ in Γ, we have $\{([\alpha_1]_{c_1} \wedge [\alpha_2]_{c_2}, \varepsilon, c_2, [\alpha_1]_{c_1}), ([\alpha_1]_{c_1} \wedge [\alpha_2]_{c_2}, \varepsilon, c_1, [\alpha_2]_{c_2})\} \subseteq \delta$, for every $[\alpha_1]_{c_1} \in Q^\gamma_\exists \setminus (\text{Axes}_\gamma \cup \text{Roots}_\gamma)$ and $[\alpha_2]_{c_2} \in \text{Triples}_\gamma \times \{\text{right}, \text{left}\} \times \{0, 1\}$.
9. For every γ in Γ, $a \in \mathbb{A}_1^\pm = \mathbb{A}_1 \dot\cup \mathbb{A}_1^{-1}$, $c \in \{0, 1\}$ and $s \in \{\text{right}, \text{left}\}$,
   - $([(L, q, p, s)]_{\max\{c,d\}}, a, d, [(L, q', p', s)]_c) \in \delta$, if there exist transitions (q, a, c_1, q') and (p', a^{-1}, c_2, p) in $\mathcal{A}_L$ such that $d = \max\{c_1, c_2\}$; and $s = \text{right} \Leftrightarrow a \in \mathbb{A}_1$. We reduce the search for the looping s-ward partial run of $\mathcal{A}_L$ from q to p, to look for a looping s-ward partial run from q' to p'.
   - For every $D = (q_1, \ldots, q_\ell)$ and $D' = (q'_1, \ldots, q'_\ell)$ in $\text{Cuts}_\gamma(x, y)$, for some variables x, y, we have $([(D, s)]_{\max\{c,d\}}, a, d, [(D', s)]_c) \in \delta$, if there are transitions (q_1, a, c_1, q'_1), $\ldots$, $(q_\ell, a, c_\ell, q'_\ell)$ in $\mathcal{A}_{L_1}, \ldots, \mathcal{A}_{L_\ell}$, respectively, where L_i is the i-th language mentioned in $\text{Atoms}_\gamma(x, y)$, such that $d = \max\{c_1, \ldots, c_\ell\}$; and $s = \text{right} \Leftrightarrow a \in \mathbb{A}_1$. These transitions correspond to a simultaneous mapping of all the atoms in $\text{Atoms}_\gamma(x, y)$ to the input expansion. In terms of special trees, these correspond to traverse one edge of the special path.
10. For γ in Γ and every final cut $D \in \text{Cuts}_\gamma(x, y)$, for some variables x, y in γ, and $s \in \{\text{right}, \text{left}\}$, we have $([(D, s)]_0, \varepsilon, 0, y) \in \delta$.
11. Finally, $(x, \varepsilon, 0, q_f) \in \delta$, for every γ in Γ and x an existentially quantified variable that is a leaf in γ with $\text{Atoms}_\gamma(x, x) = \emptyset$.

▶ **Lemma 23.** *Γ is bounded iff $\mathcal{B}$ is limited.*

Proof. Note first that every accepting run ρ of $\mathcal{B}$ on w_λ, for an expansion λ of Γ, determines a disjunct γ_ρ of Γ, an expansion λ_ρ of γ_ρ and a homomorphism h_ρ witnessing $\lambda_\rho \to \lambda$. Conversely, for every disjunct γ' of Γ, every expansion λ' of γ' and homomorphism h' witnessing $\lambda' \to \lambda$, there is an accepting run ρ of $\mathcal{B}$ on w_λ with $\gamma_\rho = \gamma'$, $\lambda_\rho = \lambda'$ and $h_\rho = h'$. Hence, it suffices to show the above-mentioned condition (2), *i.e.*,

$$\mathit{cost}(\rho) \leq \|\lambda_\rho\| \leq g(\mathit{cost}(\rho)), \text{ for every accepting run } \rho \text{ of } \mathcal{B} \text{ over } w_\lambda, \tag{2}$$

for some non-decreasing function g. In order to show this, we follow an argument similar to the one of Lemma 21. Note that every accepting run ρ of $\mathcal{B}$ on w_λ, determines a collection $C_\rho = Z_\rho \cup S_\rho$, such that $Z_\rho = \bigcup\{A_\rho : A \in \mathrm{Atoms}_{\gamma_\rho}(x,x) \text{ for some variable } x \text{ in } \gamma_\rho\}$, where A_ρ is a collection of rightward and leftward zig-zag trees; and $S_\rho = \{t_\rho^{x,y} : x \text{ is the parent of } y \text{ in } \gamma_\rho\}$, where $t_\rho^{x,y}$ is either a rightward or leftward special tree.

Let t be a (rightward or leftward) zig-zag or special tree. We write $\mathit{cost}(t)$ for the sum of all the costs, over all edges of t. For a branch B of t, a *heavy branching* of B is a subtree attached to B that has at least one edge with cost > 0. We define $f(t)$ to be the maximum over all branches B of t, of the cost of B (*i.e.*, the sum of the costs of the edges of B) plus the number of heavy branchings of B. Note that $f(t) \leq \mathit{cost}(t)$. Now let ρ be an accepting run of $\mathcal{B}$ over w_λ. We have that $\|\lambda_\rho\| = \sum_{t \in C_\rho} \mathit{cost}(t)$. By construction of $\mathcal{B}$, we have that $\mathit{cost}(\rho) \leq \sum_{t \in C_\rho} f(t)$. Then,

$$\mathit{cost}(\rho) \leq \sum_{t \in C_\rho} f(t) \leq \sum_{t \in C_\rho} \mathit{cost}(t) = \|\lambda_\rho\|,$$

which proves one of the directions of condition (2).

For the other direction, for every $t \in C_\rho$, we have that $f(t) \leq r \cdot \mathit{cost}(\rho)$, where $r := 1 + \sum\{|Q_L|^4 : L \text{ appearing in } \Gamma\}$, where Q_L is the statespace of $\mathcal{A}_L$ and r is an upper bound for the maximum arity of any zig-zag or special tree. Also, for every $t \in C_\rho$, we have $\mathit{cost}(t) \leq (2k+3)r^{f(t)}$. (Recall that k is the upper bound on the thickness of Γ.) Indeed, consider the heaviest branch B of t (*i.e.*, the result of traversing t from the root by always choosing a child whose subtree has maximal total cost). We can partition the edges of B into E_1 and E_2 such that E_1 are the edges that do not decrease the total cost of the current subtree and E_2 the ones that do. Let $n := \mathit{cost}(t)$ be the initial total cost of t. Note that each edge in E_2 decreases the total cost of the current subtree from n' to no less than $\frac{n'}{r} - (k+1)$ (note that $k+1$ is an upper bound for the cost of any edge in any zig-zag or special tree). We have that $|E_2| \geq \max\{\ell \in \mathbb{N} : \frac{n}{r^\ell} - (k+1)\sum_{i=0}^{\ell-1} \frac{1}{r^i} \geq 1\} \geq \max\{\ell \in \mathbb{N} : \frac{n}{r^\ell} - 2(k+1) \geq 1\} \geq \log_r \frac{n}{2(k+1)+1}$. The claim follows since $f(t) \geq |E_2|$ (as each edge in E_2 either has cost > 0 or has a heavy branching). Summing up, we have that

$$\|\lambda_\rho\| = \sum_{t \in C_\rho} \mathit{cost}(t) \leq \sum_{t \in C_\rho} (2k+3)r^{f(t)} \leq \sum_{t \in C_\rho} (2k+3)r^{r \cdot \mathit{cost}(\rho)} \leq r \cdot N_\Gamma (2k+3) r^{r \cdot \mathit{cost}(\rho)},$$

where N_Γ is the number of atoms of Γ (note that $r \cdot N_\Gamma$ is then an upper bound to $|C_\rho|$). This shows the remaining direction of condition (2), and hence the lemma. ◀

Finally, note that the number of states of $\mathcal{B}$ is polynomial in $\|\Gamma\|$, and hence $\mathcal{B}$ can be constructed in polynomial time. Indeed, the crucial part is to bound $|\mathrm{Cuts}_\gamma(x,y)|$ for any disjunct γ and variables x, y in γ. Since the thickness of Γ is $\leq k$, we have that $|\mathrm{Cuts}_\gamma(x,y)| \leq \|\Gamma\|^k$. This finishes the proof of Theorem 15.

Proof of Theorem 16

We start with the Π_2^P upper bound. Let Γ be a strongly connected UCRPQ. Let $\gamma(\bar{x})$ be a disjunct of Γ. We define $\gamma^{<\infty}(\bar{x})$ to be the CRPQ obtained from $\gamma(\bar{x})$ by adding an atom $x \xrightarrow{\varepsilon} x$, for each free variable x in $\bar{x}$, and removing all atoms $y \xrightarrow{L} z$ such that L is infinite. Note that $\gamma^{<\infty}(\bar{x})$ could be not well-defined. (This happens precisely when γ is Boolean and all of its RPQs are infinite.) We define $\Gamma^{<\infty} := \bigvee\{\gamma^{<\infty} : \gamma \text{ in } \Gamma, \text{ and } \gamma^{<\infty} \text{ is well-defined}\}$. If every $\gamma^{<\infty}$ is not well-defined, then $\Gamma^{<\infty}$ is not well-defined neither. We say that a UCRPQ Γ is *ε-trivial* if it has at most one free variable, and there is a CRPQ γ in Γ such that all of its RPQs contain the empty word ε. Note that an ε-trivial UCRPQ is always bounded. We have the following:

▶ **Lemma 24.** *A strongly connected UCRPQ Γ is bounded iff Γ is ε-trivial or, $\Gamma^{<\infty}$ is well-defined and $\Gamma^{<\infty} \subseteq \Gamma$.*

Proof. From right to left, if Γ is ε-trivial then it is bounded. Otherwise, $\Gamma^{<\infty} \subseteq \Gamma$ and then $\Gamma^{<\infty}$ is equivalent to Γ (as $\Gamma \subseteq \Gamma^{<\infty}$ always holds). Since $\Gamma^{<\infty}$ is bounded, then Γ is also bounded. From left to right, suppose Γ is bounded and not ε-trivial. By Proposition 3, there is $k \geq 1$ such that for every expansion λ of Γ there is an expansion λ' of Γ such that $\|\lambda'\| \leq k$ and $\lambda' \to \lambda$ (†). We show first that $\Gamma^{<\infty}$ is well-defined. By contradiction suppose this is not the case. In particular, all the RPQs in Γ are infinite. We pick an arbitrary disjunct γ of Γ and an expansion $\lambda_{>k}$ of γ obtained from choosing a word $w \in L$ with $|w| > k$, for every atom $x \xrightarrow{L} y$ of γ. By (†), there is an expansion λ' such that $\|\lambda'\| \leq k$ and $\lambda' \to \lambda_{>k}$. Since Γ is not ε-trivial, it follows that there at least one (non-equality) atom $x \xrightarrow{a} y$ in λ'. Since γ is strongly connected, λ' has a (labeled) directed cycle containing $x \xrightarrow{a} y$ (*i.e.*, number of edges) at most k. Since every directed cycle in $\lambda_{>k}$ has length greater than k, we have a contradiction with the fact that $\lambda' \to \lambda_{>k}$.

Now we show $\Gamma^{<\infty} \subseteq \Gamma$ using Lemma 17. Let λ be any expansion of $\gamma^{<\infty}$ in $\Gamma^{<\infty}$. If $\gamma = \gamma^{<\infty}$, then we are done. Otherwise, consider the expansion $\lambda_{>k}$ of γ obtained by (1) choosing the same word as in λ for atoms $x \xrightarrow{L} y$ with L finite, and (1) choosing a word $w \in L$ such that $|w| > k$ for the atoms $x \xrightarrow{L} y$ with L infinite. Note that we can partition the (non-equality) atoms of $\lambda_{>k}$ into those generated in case (1), denoted by $A^{<\infty}$ and those generated in case (2), denoted by A^{∞}. By (†), there is an expansion λ' of Γ with $\|\lambda'\| \leq k$ such that $\lambda' \to \lambda_{>k}$ via a homomorphism h. We claim that the image via h of every atom $x \xrightarrow{a} y$ in λ' belongs to $A^{<\infty}$. By contradiction, suppose $h(x) \xrightarrow{a} h(y) \in A^{\infty}$. Since γ is strongly connected, λ' has a (labeled) directed cycle containing $x \xrightarrow{a} y$ of length $\leq k$. This is a contradiction as $\lambda' \to \lambda_{>k}$ and every directed cycle in $\lambda_{>k}$ has length $> k$. Hence, $\lambda' \to \lambda$. By Lemma 17, we obtain that $\Gamma^{<\infty} \subseteq \Gamma$. ◀

For the lower bound, we reduce from the following well-known Π_2^P-complete problem: Given a connected (undirected) graph $G = (V, E)$ and $k \geq 1$ (given in unary), check whether for every function $c : V \to \{0, 1\}$, there is a clique K in G of size k such that $c(u) = c(v)$ for all nodes u, v in K. (Recall that a clique is a graph with an edge between each pair of distinct nodes.)

Given $G = (V, E)$ and $k \geq 1$, we define a Boolean strongly connected UCRPQ Γ over the alphabet $\mathbb{A} := \{a, b, 0, 1\}$ as follows. Let γ_G be the Boolean CRPQ with variable set $\{x_u : u \in V\}$ where we have atoms $x_u \xrightarrow{a} x_v, x_u \xrightarrow{b} x_v, x_v \xrightarrow{a} x_u, x_v \xrightarrow{b} x_u$, for each edge $\{u, v\} \in E$, and an atom $x_u \xrightarrow{0+1} x_u$, for each node $u \in V$. For $\ell \in \{0, 1\}$, we define the Boolean CRPQ γ_k^ℓ to have variable set $\{z_0, \ldots, z_{k-1}\}$, atoms $z_i \xrightarrow{a} z_j, z_j \xrightarrow{a} z_i$, for each pair

$i \neq j \in \{0, \ldots, k-1\}$, and an atom $z_i \xrightarrow{\ell} z_i$, for each $i \in \{0, \ldots, k-1\}$. We pick an arbitrary node u_0 from G and for every $\ell \in \{0,1\}$, we define γ_G^ℓ to be the Boolean CRPQ obtained from the (disjoint) conjunction of γ_G and γ_k^ℓ by adding the atoms $x_{u_0} \xrightarrow{b^*} z_0, z_0 \xrightarrow{b^*} x_{u_0}$. Then we let $\Gamma := \gamma_G^0 \vee \gamma_G^1$. Note that Γ is actually strongly connected. Also observe that $\Gamma^{<\infty} = (\gamma_G^0)^{<\infty} \vee (\gamma_G^1)^{<\infty}$, where $(\gamma_G^\ell)^{<\infty}$ is the (disjoint) conjunction of γ_G and γ_k^ℓ, for $\ell \in \{0,1\}$.

We claim that G, k is a positive instance iff Γ is bounded. We show that G, k is a positive instance iff $\Gamma^{<\infty} \subseteq \Gamma$, a hence the claim follows from Lemma 24. Suppose that G, k is a positive instance. We prove $\Gamma^{<\infty} \subseteq \Gamma$ using Lemma 17. Let λ be an any expansion of $\Gamma^{<\infty}$. Then there is $\ell \in \{0,1\}$ and a function $c : V \to \{0,1\}$ such that λ is the disjoint conjunction of γ_G^c and γ_k^ℓ, where γ_G^c is obtained from γ_G by replacing $x_u \xrightarrow{0+1} x_u$ with $x_u \xrightarrow{c(u)} x_u$, for each $u \in V$. By hypothesis, G contains a clique K of size k with $c(z) = \ell'$, for each z in K and some fixed $\ell' \in \{0,1\}$. Pick an arbitrary node z^* of K and consider the expansion λ' of $\gamma_G^{\ell'}$ given by $\lambda_G^c \wedge \gamma_k^{\ell'} \wedge x_{u_0} \xrightarrow{b^d} z_0 \wedge z_0 \xrightarrow{b^d} x_{u_0}$, where $d \geq 0$ is the distance in G from u_0 to z^*. Then $\lambda' \to \lambda_G^c \to \lambda$ via the homomorphism h that is the identity over λ_G^c and maps $\gamma_k^{\ell'}$ to $\{x_z : z \text{ in } K\}$ with $h(z_0) = x_{z^*}$. Hence, $\Gamma^{<\infty} \subseteq \Gamma$.

Suppose now that $\Gamma^{<\infty} \subseteq \Gamma$ and let c be any function $c : V \to \{0,1\}$. Consider the expansion λ_c of γ_G^0 given by $\gamma_G^c \wedge \gamma_k^0$, where γ_G^c is defined as above. By Lemma 17, there is an expansion λ_ℓ of some γ_G^ℓ with $\ell \in \{0,1\}$, such that $\lambda_\ell \to \lambda_c$. Since λ_ℓ is connected, and contains symbols $b \in \mathbb{A}$, it is the case that $\lambda_\ell \to \gamma_G^c$ via h. Hence $\{h(z_i) : i \in \{0, \ldots, k-1\}\}$ must correspond to a clique K of G with $c(z) = \ell$, for all z in K.