A novel family of finite automata for recognizing and learning ω-regular languages

Yong Li, Sven Schewe, Qiyi Tang

University of Liverpool, UK

Abstract. Families of DFAs (FDFAs) have recently been introduced as a new representation of ω-regular languages. They target ultimately periodic words, with acceptors revolving around accepting some representation $u \cdot v^\omega$. Three canonical FDFAs have been suggested, called *periodic*, *syntactic*, and *recurrent*. We propose a fourth one, *limit FDFAs*, which can be exponentially coarser than periodic FDFAs and are more succinct than syntactic FDFAs, while they are incomparable (and dual to) recurrent FDFAs. We show that limit FDFAs can be easily used to check not only whether ω-languages are regular, but also whether they are accepted by deterministic Büchi automata. We also show that canonical forms can be left behind in applications: the limit and recurrent FDFAs can complement each other nicely, and it may be a good way forward to use a combination of both. Using this observation as a starting point, we explore making more efficient use of Myhill-Nerode's right congruences in aggressively increasing the number of don't-care cases in order to obtain smaller progress automata. In pursuit of this goal, we gain succinctness, but pay a high price by losing constructiveness.

1 Introduction

The class of ω-regular languages has proven to be an important formalism to model reactive systems and their specifications, and automata over infinite words are the main tool to reason about them. For example, the automata-theoretic approach to verification [25] is the main framework for verifying ω-regular specifications. The first type of automata recognizing ω-regular languages is nondeterministic Büchi automata [6] (NBAs) where an infinite word is accepted if one of its runs meets the accepting condition for infinitely many times. Since then, other types of acceptance conditions, such as Muller, Rabin, Streett and parity automata [26], have been introduced. All the automata mentioned above are finite automata processing *infinite* words, widely known as ω-automata [26].

The theory of ω-regular languages is more involved than that of regular languages. For instance, nondeterministic finite automata (NFAs) can be determinized with a subset construction, while NBAs have to make use of tree structures [22]. This is because of a fundamental difference between these language classes: for a given regular language R, the Myhill-Nerode theorem [19,20] defines a right congruence (RC) $\backsim_R$ in which every equivalence class corresponds to a state in the minimal deterministic finite automata (DFA) accepting R. In contrast, there is no similar theorem to define the minimal deterministic ω-automata for the full class of ω-regular languages[1]. Schewe proved in [24] that it is NP-complete to find the minimal deterministic ω-automaton even given a deterministic ω-automaton. Therefore, it seems impossible to easily define a Myhill-Nerode theorem for (minimal) ω-automata.

Recently, Angluin, Boker and Fisman [2] proposed families of DFAs (FDFAs) for recognizing ω-regular languages, in which every DFA can be defined with respect to a RC defined over a given ω-regular language [3]. This tight connection is the theoretical foundation on which the state of the art learning algorithms for ω-regular languages [3, 13] using membership and equivalence queries [1] are built. FDFAs are based on well-known properties of ω-regular languages [6, 7]: two ω-regular languages are equivalent if, and only if, they have the same set of *ultimately periodic words*. An ultimately periodic word w is an infinite word that consists of first a finite prefix u, followed by an infinite repetition of a finite nonempty word v; it can thus be represented as a decomposition pair (u, v). FDFAs accept infinite words by accepting their decomposition pairs: an FDFA $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\})$ consists of a *leading DFA* $\mathcal{M}$ that processes the finite prefix u, while leaving the acceptance work of v to the *progress DFA* $\mathcal{N}^q$, one for each state of $\mathcal{M}$. To this end, $\mathcal{M}$ intuitively tracks the Myhill-Nerode's RCs, and an ultimately periodic word $u \cdot v^\omega$ is accepted if it has a representation $x \cdot y^\omega$ such that x and $x \cdot y$ are in the same congruence class and y is accepted by the progress DFA $\mathcal{N}^x$. Angluin and Fisman [3] formalized the RCs of three canonical FDFAs, namely periodic [7], syntactic [17] and recurrent [3], and provided a unified learning framework for them.

In this work, we first propose a fourth one, called *limit FDFAs* (cf. Section 3). We show that limit FDFAs are coarser than syntactic FDFAs. Since syntactic FDFAs can be exponentially more succinct than periodic FDFAs [3], so do our limit FDFAs. We show that limit FDFAs are dual (and thus incomparable in the size) to recurrent FDFAs, due to symmetric treatment for don't care words. More precisely, the formalization of such FDFA does not care whether or not a progress automaton $\mathcal{N}^x$ accepts or rejects a word v, unless reading it in $\mathcal{M}$ produces a self-loop. Recurrent progress DFAs reject all those don't care words, while limit progress DFAs accept them.

We show that limit FDFAs (families of DFAs that use limit DFAs) have two interesting properties. The first is on conciseness: we show that this change in the treatment of don't care words not only defines a dual to recurrent FDFAs but also allows us to identify languages accepted by deterministic Büchi automata (DBAs) easily. It is only known that one can identify whether a given ω-language is regular by verifying whether the number of states in the three canonical FDFAs is finite. However, if one wishes to identify DBA-recognizable languages with FDFAs, a straight-forward approach is to first translate the input FDFA to an equivalent deterministic Rabin automaton [2] through an intermediate NBA, and then use the deciding algorithm in [11] by checking the transition structure

[1] Simple extension of Myhill-Nerode theorem for ω-regular languages only works on a small subset [4, 16]

of Rabin automata. However, this approach is exponential in the size of the input FDFA because of the NBA determinization procedure [8,22,23]. Our limit FDFAs are, to the best of our knowledge, the *first* type of FDFAs able to identify the DBA-recognizable languages in polynomial time (cf. Section 4).

We note that limit FDFAs also fit nicely into the learning framework introduced in [3], so that they can be used for learning without extra development.

We then discuss how to make more use of don't care words when defining the RCs of the progress automata, leading to the coarsest congruence relations and therefore the most concise FDFAs, albeit to the expense of losing constructiveness (cf. Section 5).

2 Preliminaries

In the whole paper, we fix a finite *alphabet* Σ. A *word* is a finite or infinite sequence of letters in Σ; ϵ denotes the empty word. Let Σ^* and Σ^ω denote the set of all finite and infinite words (or ω-words), respectively. In particular, we let $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. A *finitary language* is a subset of Σ^*; an ω*-language* is a subset of Σ^ω. Let ρ be a sequence; we denote by $\rho[i]$ the i-th element of ρ and by $\rho[i..k]$ the subsequence of ρ starting at the i-th element and ending at the k-th element (inclusively) when $i \leq k$, and the empty sequence ϵ when $i > k$. Given a finite word u and a word w, we denote by $u \cdot w$ (uw, for short) the concatenation of u and w. Given a finitary language L_1 and a finitary/ω-language L_2, the concatenation $L_1 \cdot L_2$ (L_1L_2, for short) of L_1 and L_2 is the set $L_1 \cdot L_2 = \{ uw \mid u \in L_1, w \in L_2 \}$ and L_1^ω the infinite concatenation of L_1.

Transition system. A (nondeterministic) transition system (TS) is a tuple $\mathcal{T} = (Q, q_0, \delta)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma \to 2^Q$ is a transition function. We also lift δ to sets as $\delta(S, \sigma) := \bigcup_{q \in S} \delta(q, \sigma)$. We also extend δ to words, by letting $\delta(S, \epsilon) = S$ and $\delta(S, a_0a_1 \cdots a_k) = \delta(\delta(S, a_0), a_1, \cdots, a_k)$, where we have $k \geq 1$ and $a_i \in \Sigma$ for $i \in \{0, \cdots, k\}$.

The *underlying graph* $\mathcal{G}_\mathcal{T}$ of a TS $\mathcal{T}$ is a graph $\langle Q, E \rangle$, where the set of vertices is the set Q of states in $\mathcal{T}$ and $(q, q') \in E$ if $q' \in \delta(q, a)$ for some $a \in \Sigma$. We call a set $C \subseteq Q$ a *strongly connected component* (SCC) of $\mathcal{T}$ if, for every pair of states $q, q' \in C$, q and q' can reach each other in $\mathcal{G}_\mathcal{T}$.

Automata. An automaton on finite words is called a *nondeterministic finite automaton* (NFA). An NFA $\mathcal{A}$ is formally defined as a tuple $(\mathcal{T}, F)$, where $\mathcal{T}$ is a TS and $F \subseteq Q$ is a set of *final* states. An automaton on ω-words is called a *nondeterministic Büchi automaton* (NBA). An NBA $\mathcal{B}$ is represented as a tuple $(\mathcal{T}, \Gamma)$ where $\mathcal{T}$ is a TS and $\Gamma \subseteq \{(q, a, q') : q, q' \in Q, a \in \Sigma, q' \in \delta(q, a)\}$ is a set of *accepting* transitions. An NFA $\mathcal{A}$ is said to be a *deterministic* finite automaton (DFA) if, for each $q \in Q$ and $a \in \Sigma$, $|\delta(q, a)| \leq 1$. Deterministic Büchi automata (DBAs) are defined similarly and thus Γ is a subset of $\{(q, a) : q \in Q, a \in \Sigma\}$, since the successor q' is determined by the source state and the input letter.

A *run* of an NFA $\mathcal{A}$ on a finite word u of length $n \geq 0$ is a sequence of states $\rho = q_0 q_1 \cdots q_n \in Q^+$ such that, for every $0 \leq i < n$, $q_{i+1} \in \delta(q_i, u[i])$. We write $q_0 \xrightarrow{u} q_n$ if there is a run from q_0 to q_n over u. A finite word $u \in \Sigma^*$ is *accepted* by an NFA $\mathcal{A}$ if there is a run $q_0 \cdots q_n$ over u such that $q_n \in F$. Similarly, an ω-*run* of $\mathcal{A}$ on an ω-word w is an infinite sequence of transitions $\rho = (q_0, w[0], q_1)(q_1, w[1], q_2) \cdots$ such that, for every $i \geq 0$, $q_{i+1} \in \delta(q_i, w[i])$. Let $\inf(\rho)$ be the set of transitions that occur infinitely often in the run ρ. An ω-word $w \in \Sigma^\omega$ is *accepted* by an NBA $\mathcal{A}$ if there exists an ω-run ρ of $\mathcal{A}$ over w such that $\inf(\rho) \cap \Gamma \neq \emptyset$. The *finitary language* recognized by an NFA $\mathcal{A}$, denoted by $\mathcal{L}_*(\mathcal{A})$, is defined as the set of finite words accepted by it. Similarly, we denote by $\mathcal{L}(\mathcal{A})$ the ω-*language* recognized by an NBA $\mathcal{A}$, i.e., the set of ω-words accepted by $\mathcal{A}$. NFAs/DFAs accept exactly *regular* languages while NBAs recognize exactly ω-*regular* languages.

Right congruences. A *right congruence* (RC) relation is an equivalence relation $\backsim$ over Σ^* such that $x \backsim y$ implies $xv \backsim yv$ for all $v \in \Sigma^*$. We denote by $|\backsim|$ the index of $\backsim$, i.e., the number of equivalence classes of $\backsim$. A *finite RC* is a RC with a finite index. We denote by $\Sigma^*/_\backsim$ the set of equivalence classes of Σ^* under $\backsim$. Given $x \in \Sigma^*$, we denote by $[x]_\backsim$ the equivalence class of $\backsim$ that x belongs to.

For a given RC $\backsim$ of a regular language R, the Myhill-Nerode theorem [19,20] defines a unique minimal DFA D of R, in which each state of D corresponds to an equivalence class defined by $\backsim$ over Σ^*. Therefore, we can construct a DFA $\mathcal{D}[\backsim]$ from $\backsim$ in a standard way.

Definition 1 ([19, 20]). *Let $\backsim$ be a right congruence of finite index. The TS $\mathcal{T}[\backsim]$ induced by $\backsim$ is a tuple (S, s_0, δ) where $S = \Sigma^*/_\backsim$, $s_0 = [\epsilon]_\backsim$, and for each $u \in \Sigma^*$ and $a \in \Sigma$, $\delta([u]_\backsim, a) = [ua]_\backsim$.*

For a given regular language R, we can define the RC $\backsim_R$ of R as $x \backsim_R y$ if, and only if, $\forall v \in \Sigma^*.\ xv \in R \iff yv \in R$. Therefore, the minimal DFA for R is the DFA $\mathcal{D}[\backsim_R] = (\mathcal{T}[\backsim_R], F_{\backsim_R})$ by setting final states $F_{\backsim_R}$ to all equivalence classes $[u]_{\backsim_R}$ such that $u \in R$.

Ultimately periodic (UP) words. A UP-word w is an ω-word of the form uv^ω, where $u \in \Sigma^*$ and $v \in \Sigma^+$. Thus $w = uv^\omega$ can be represented as a pair of finite words (u, v), called a *decomposition* of w. A UP-word can have multiple decompositions: for instance (u, v), (uv, v), and (u, vv) are all decompositions of uv^ω. For an ω-language L, let $\mathrm{UP}(L) = \{ uv^\omega \in L \mid u \in \Sigma^* \wedge v \in \Sigma^+ \}$ denote the set of all UP-words in L. The set of UP-words of an ω-regular language L can be seen as the fingerprint of L, as stated below.

Theorem 1 ([6, 7]). *(1) Every non-empty ω-regular language L contains at least one UP-word. (2) Let L and L' be two ω-regular languages. Then $L = L'$ if, and only if, $UP(L) = UP(L')$.*

Families of DFAs (FDFAs). Based on Theorem 1, Angluin, Boker, and Fisman [2] introduced the notion of FDFAs to recognize ω-regular languages.

Definition 2 ([2]). *An FDFA is a pair $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\})$ consisting of a leading DFA $\mathcal{M}$ and of a progress DFA $\mathcal{N}^q$ for each state q in $\mathcal{M}$.*

Intuitively, the leading DFA $\mathcal{M}$ of $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\})$ for L consumes the finite prefix u of a UP-word $uv^\omega \in \mathrm{UP}(L)$, reaching some state q and, for each state q of $\mathcal{M}$, the progress DFA $\mathcal{N}^q$ accepts the period v of uv^ω. Note that the leading DFA $\mathcal{M}$ of every FDFA does not make use of final states—contrary to its name, it is really a leading transition system.

Let A be a deterministic automaton with TS $\mathcal{T} = (Q, q_0, \delta)$ and $x \in \Sigma^*$. We denote by $A(x)$ the state $\delta(q_0, x)$. Each FDFA $\mathcal{F}$ characterizes a set of UP-words $\mathrm{UP}(\mathcal{F})$ by following the acceptance condition.

Definition 3 (Acceptance). *Let $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\})$ be an FDFA and w be a UP-word. A decomposition (u, v) of w is* normalized *with respect to $\mathcal{F}$ if $\mathcal{M}(u) = \mathcal{M}(uv)$. A decomposition (u, v) is accepted by $\mathcal{F}$ if (u, v) is normalized and we have $v \in \mathcal{L}_*(\mathcal{N}^q)$ where $q = \mathcal{M}(u)$. The UP-word w is accepted by $\mathcal{F}$ if there exists a decomposition (u, v) of w accepted by $\mathcal{F}$.*

Note that the acceptance condition in [2] is defined with respect to the decompositions, while ours applies to UP-words. So, they require the FDFAs to be saturated for recognizing ω-regular languages.

Definition 4 (Saturation [2]). *Let $\mathcal{F}$ be an FDFA and w be a UP-word in $UP(\mathcal{F})$. We say $\mathcal{F}$ is* saturated *if, for all normalized decompositions (u, v) and (u', v') of w, either both (u, v) and (u', v') are accepted by $\mathcal{F}$, or both are not.*

We will see in Section 4.1 that under our acceptance definition the saturation property can be relaxed while still accepting the same language.

In the remainder of the paper, we fix an ω-language L unless stated otherwise.

3 Limit FDFAs for recognizing ω-regular languages

In this section, we will first recall the definitions of three existing canonical FDFAs for ω-regular languages, and then introduce our limit FDFAs and compare the four types of FDFAs.

3.1 Limit FDFAs and other canonical FDFAs

Recall that, for a given regular language R, by Definition 1, the Myhill-Nerode theorem [19, 20] associates each equivalence class of $\backsim_R$ with a state of the minimal DFA $\mathcal{D}[\backsim_R]$ of R. The situation in ω-regular languages is, however, more involved [4]. An immediate extension of such RCs for an ω-regular language L is the following.

Definition 5 (Leading RC). *For two $u_1, u_2 \in \Sigma^*$, $u_1 \backsim_L u_2$ if, and only if $\forall w \in \Sigma^\omega.\ u_1 w \in L \Longleftrightarrow u_2 w \in L$.*

Since we fix an ω-language L in the whole paper, we will omit the subscript in $\backsim_L$ and directly use $\backsim$ in the remainder of the paper.

Assume that L is an ω-regular language. Obviously, the index of $\backsim$ is *finite* since it is not larger than the number of states in the minimal deterministic ω-automaton accepting L. However, $\backsim$ is only enough to define the minimal ω-automaton for a small subset of ω-regular languages; see [4,16] for details about such classes of languages. For instance, consider the language $L = (\Sigma^* \cdot aa)^\omega$ over $\Sigma = \{a, b\}$: clearly, $|\backsim| = 1$ because L is a suffix language (for all $u \in \Sigma^*$, $w \in L \iff u \cdot w \in L$). At the same time, it is easy to see that the minimal deterministic ω-automaton needs at least two states to recognize L. Hence, $\backsim$ alone does not suffice to recognize the full class of ω-regular languages.

Nonetheless, based on Theorem 1, we only need to consider the UP-words when uniquely identifying a given ω-regular language L with RCs. Calbrix *et al.* proposed in [7] the use of the regular language $L_\$ = \{u\$v : u \in \Sigma^*, v \in \Sigma^+, uv^\omega \in L\}$ to represent L, where $\$ \notin \Sigma$ is a fresh letter[2]. Intuitively, $L_\$$ associates a UP-word w in $\mathrm{UP}(L)$ by containing every decomposition (u, v) of w in the form of $u\$v$. The FDFA representing $L_\$$ is formally stated as below.

Definition 6 (Periodic FDFAs [7]). *The $\backsim$ is as defined in Definition 5.*

Let $[u]_\backsim$ be an equivalence class of $\backsim$. For $x, y \in \Sigma^*$, we define periodic RC as: $x \approx_P^u y$ if, and only if, $\forall v \in \Sigma^*$, $u \cdot (x \cdot v)^\omega \in L \iff u \cdot (y \cdot v)^\omega \in L$.

The periodic FDFA $\mathcal{F}_P = (\mathcal{M}, \{\mathcal{N}_P^u\})$ of L is defined as follows.

The leading DFA $\mathcal{M}$ is the tuple $(\mathcal{T}[\backsim], \emptyset)$. Recall that $\mathcal{T}[\backsim]$ is the TS constructed from $\backsim$ by Definition 1.

The periodic progress DFA $\mathcal{N}_P^u$ of the state $[u]_\backsim \in \Sigma^*/_\backsim$ is the tuple $(\mathcal{T}[\approx_P^u], F_u)$, where $[v]_{\approx_P^u} \in F_u$ if $uv^\omega \in L$.

One can verify that, for all $u, x, y, v \in \Sigma^*$, if $x \approx_P^u y$, then $xv \approx_P^u yv$. Hence, $\approx_P^u$ is a RC. It is also proved in [7] that $L_\$$ is a regular language, so the index of $\approx_P^u$ is also finite.

Angluin and Fisman in [3] showed that, for a variant of the family of languages L_n given by Michel [18], its periodic FDFA has $\Omega(n!)$ states, while the syntactic FDFA obtained in [17] only has $\mathcal{O}(n^2)$ states. The leading DFA of the syntactic FDFAs is exactly the one defined for the periodic FDFA. The two types of FDFAs differ in the definitions of the progress DFAs $\mathcal{N}^u$ for some $[u]_\backsim$. From Definition 6, one can see that $\mathcal{N}_P^u$ accepts the finite words in $V_u = \{v \in \Sigma^+ : u \cdot v^\omega \in L\}$. The progress DFA $\mathcal{N}_S^u$ of the syntactic FDFA is not required to accept all words in V_u, but only a subset $V_{u,v} = \{v \in \Sigma^+ : u \cdot v^\omega \in L, u \backsim u \cdot v\}$, over which the leading DFA $\mathcal{M}$ can take a round trip from $\mathcal{M}(u)$ back to itself. This minor change makes the syntactic FDFAs of the language family L_n exponentially more succinct than their periodic counterparts.

Formally, syntactic FDFAs are defined as follows.

Definition 7 (Syntactic FDFA [17]). *The $\backsim$ is as defined in Definition 5.*

[2] This enables to learn L via learning the regular language $L_\$$ [10].

Let $[u]_{\backsim}$ be an equivalence class of $\backsim$. For $x, y \in \Sigma^*$, we define syntactic RC as: $x \approx_S^u y$ if and only if $u \cdot x \backsim u \cdot y$ and for $\forall v \in \Sigma^*$, if $u \cdot x \cdot v \backsim u$, then $u \cdot (x \cdot v)^\omega \in L \Longleftrightarrow u \cdot (y \cdot v)^\omega \in L$.

The syntactic FDFA $\mathcal{F}_S = (\mathcal{M}, \{\mathcal{N}_S^u\})$ of L is defined as follows.

The leading DFA $\mathcal{M}$ is the tuple $(\mathcal{T}[\backsim], \emptyset)$ as defined in Definition 6.

The syntactic progress DFA $\mathcal{N}_S^u$ of the state $[u]_{\backsim} \in \Sigma^*/_{\backsim}$ is the tuple $(\mathcal{T}[\approx_S^u], F_u)$ where $[v]_{\approx_S^u} \in F_u$ if $u \cdot v \backsim u$ and $uv^\omega \in L$.

Angluin and Fisman [3] noticed that the syntactic progress RCs are not defined with respect to the regular language $V_{u,v} = \{v \in \Sigma^+ : u{\cdot}v^\omega \in L, u \backsim u{\cdot}v\}$ as $\backsim_{V_{u,v}}$ that is similar to $\backsim_R$ for a regular language R. They proposed the recurrent progress RC $\approx_R^u$ that mimics the RC $\backsim_{V_{u,v}}$ to obtain a DFA accepting $V_{u,v}$ as follows.

Definition 8 (Recurrent FDFAs [3]). *The $\backsim$ is as defined in Definition 5.*

Let $[u]_{\backsim}$ be an equivalence class of $\backsim$. For $x, y \in \Sigma^*$, we define recurrent RC as: $x \approx_R^u y$ if and only if $\forall v \in \Sigma^*$, $(u \cdot x \cdot v \backsim u \wedge u \cdot (xv)^\omega \in L) \Longleftrightarrow (u \cdot yv \backsim u \wedge u \cdot (y \cdot v)^\omega \in L)$.

The recurrent FDFA $\mathcal{F}_R = (\mathcal{M}, \{\mathcal{N}_R^u\})$ of L is defined as follows.

The leading DFA $\mathcal{M}$ is the tuple $(\mathcal{T}[\backsim], \emptyset)$ as defined in Definition 6.

The recurrent progress DFA $\mathcal{N}_R^u$ of the state $[u]_{\backsim} \in \Sigma^*/_{\backsim}$ is the tuple $(\mathcal{T}[\approx_R^u], F_u)$ where $[v]_{\approx_R^u} \in F_u$ if $u \cdot v \backsim u$ and $uv^\omega \in L$.

As pointed out in [3], the recurrent FDFAs may *not* be minimal because, according to Definition 3, FDFAs only care about the normalized decompositions, i.e, whether a word in $C_u = \{v \in \Sigma^+ : u \cdot v \backsim u\}$ is accepted by the progress DFA $\mathcal{N}_R^u$. However, there are *don't care* words that are not in C_u and recurrent FDFAs treat them all as *rejecting*[3].

Our argument is that the don't care words are *not* necessarily rejecting and can also be regarded as *accepting*. This idea allows the progress DFAs $\mathcal{N}^u$ to accept the regular language $\{v \in \Sigma^+ : u \cdot v \backsim u \implies u \cdot v^\omega \in L\}$, rather than $\{v \in \Sigma^+ : u \cdot v \backsim u \wedge u \cdot v^\omega \in L\}$. This change allows a translation of limit FDFAs to DBAs with a quadratic blow-up when L is DBA-recognizable language, as shown later in Section 4. We formalize this idea as below and define a new type of FDFAs called *limit FDFAs*.

Definition 9 (Limit FDFAs). *The $\backsim$ is as defined in Definition 5.*

Let $[u]_{\backsim}$ be an equivalence class of $\backsim$. For $x, y \in \Sigma^*$, we define limit RC as: $x \approx_L^u y$ if and only if $\forall v \in \Sigma^*$, $(u \cdot x \cdot v \backsim u \Longrightarrow u \cdot (x \cdot v)^\omega \in L) \Longleftrightarrow (u \cdot y \cdot v \backsim u \Longrightarrow u \cdot (y \cdot v)^\omega \in L)$.

The limit FDFA $\mathcal{F}_L = (\mathcal{M}, \{\mathcal{N}_L^u\})$ of L is defined as follows.

The leading DFA $\mathcal{M}$ is the tuple $(\mathcal{T}[\backsim], \emptyset)$ as defined in Definition 6.

The progress DFA $\mathcal{N}_L^u$ of the state $[u]_{\backsim} \in \Sigma^*/_{\backsim}$ is the tuple $(\mathcal{T}[\approx_L^u], F_u)$ where $[v]_{\approx_L^u} \in F_u$ if $u \cdot v \backsim u \implies uv^\omega \in L$.

[3] Minimizing DFAs with don't care words is NP-complete [21]

We need to show that $\approx^u_L$ is a RC. For $u, x, y, v' \in \Sigma^*$, if $x \approx^u_L y$, we need to prove that $xv' \approx^u_L yv'$, i.e., for all $e \in \Sigma^*$, $(u \cdot xv' \cdot e \backsim u \implies u \cdot (xv' \cdot e)^\omega \in L) \iff (u \cdot yv' \cdot e \backsim u \implies u \cdot (yv' \cdot e)^\omega \in L)$. This follows immediately from the fact that $x \approx^u_L y$ by setting $v = v' \cdot e$ for all $e \in \Sigma^*$ in Definition 9.

Let $L = a^\omega + ab^\omega$ be a language over $\Sigma = \{a, b\}$. Three types of FDFAs are depicted in Figure 1, where the leading DFA $\mathcal{M}$ is given in the column labeled with "Leading" and the progress DFAs are in the column labeled with "Syntactic", "Recurrent" and "Limit". We omit the periodic FDFA here since we will focus more on the other three in this work. Consider the progress DFA $\mathcal{N}^{aa}_L$: there are only two equivalence classes, namely $[\epsilon]_{\approx^{aa}_L}$ and $[a]_{\approx^{aa}_L}$. We can use $v = \epsilon$ to distinguish ϵ and a word $x \in \Sigma^+$ since $aa \cdot \epsilon \backsim aa \implies aa \cdot (\epsilon \cdot \epsilon)^\omega \in L$ does not hold, while $aa \cdot x \backsim aa \implies aa \cdot (x \cdot \epsilon)^\omega \in L$ holds. For all $x, y \in \Sigma^+$, $x \approx^{aa}_L y$ since both $aa \cdot x \backsim aa \implies aa \cdot (x \cdot v)^\omega \in L$ and $aa \cdot y \backsim aa \implies aa \cdot (y \cdot v)^\omega \in L$ hold for all $v \in \Sigma^*$. One can also verify the constructions for the syntactic and recurrent progress DFAs. We can see that the don't care word b for the class $[aa]_\backsim$ are rejecting in both $\mathcal{N}^{aa}_S$ and $\mathcal{N}^{aa}_R$, while it is accepted by $\mathcal{N}^{aa}_L$. Even though b is accepted in $\mathcal{N}^{aa}_L$, one can observe that (aa, b) (and thus $aa \cdot b^\omega$) is not accepted by the limit FDFA, according to Definition 3. Indeed, the three types of FDFAs still recognize the same language L.

When the index of $\backsim$ is only one, then $\epsilon \backsim u$ holds for all $u \in \Sigma^*$. Corollary 1 follows immediately.

Corollary 1. *Let L be an ω-regular language with $|\backsim| = 1$. Then, periodic, syntactic, recurrent and limit FDFAs coincide.*

We show in Lemma 1 that the limit FDFAs are a coarser representation of L than the syntactic FDFAs. Moreover, there is a tight connection between the syntactic FDFAs and limit FDFAs.

Lemma 1. *For all $u, x, y \in \Sigma^*$,*

1. *$x \approx^u_S y$ if, and only if $u \cdot x \backsim u \cdot y$ and $x \approx^u_L y$.*
2. *$|\approx^u_L| \leq |\approx^u_S| \leq |\backsim| \cdot |\approx^u_L|$; $|\approx^u_L| \leq |\backsim| \cdot |\approx^u_P|$.*

Proof. 1. – Assume that $ux \backsim uy$ and $x \approx^u_L y$. Since $x \approx^u_L y$ holds, then for all $v \in \Sigma^*$, $(uxv \backsim u \implies u \cdot (xv)^\omega \in L) \iff (uyv \backsim u \implies u \cdot (yv)^\omega \in L)$. Since $ux \backsim uy$ holds, then $u \cdot xv \backsim u \iff u \cdot yv \backsim u$ for all $v \in \Sigma^*$. Hence, by Definition 7, if $uxv \not\backsim u$ (and thus $uyv \not\backsim u$), it follows that $x \approx^u_S y$ by definition of $\approx^u_S$; otherwise we have both $uxv \backsim u$ and $uyv \backsim u$ hold, and also $u \cdot (xv)^\omega \in L \iff u \cdot (yv)^\omega \in L$, following the definition of $\approx^u_L$. It thus follows that $x \approx^u_S y$.

– Assume that $x \approx^u_S y$. First, we have $ux \backsim uy$ by definition of $\approx^u_S$. Since $ux \backsim uy$ holds, then $u \cdot xv \backsim u \iff u \cdot yv \backsim u$ for all $v \in \Sigma^*$. Assume by contradiction that $x \approx^u_L y$. Then there must exist some $v \in \Sigma^*$ such that $u \cdot xv \backsim u \cdot yv \backsim u$ holds but $u \cdot (xv)^\omega \in L \iff u \cdot (yv)^\omega \in L$ does not hold. By definition of $\approx^u_S$, it then follows that $x \not\approx^S_u y$, violating our assumption. Hence, both $ux \backsim uy$ and $x \approx^u_L y$ hold.

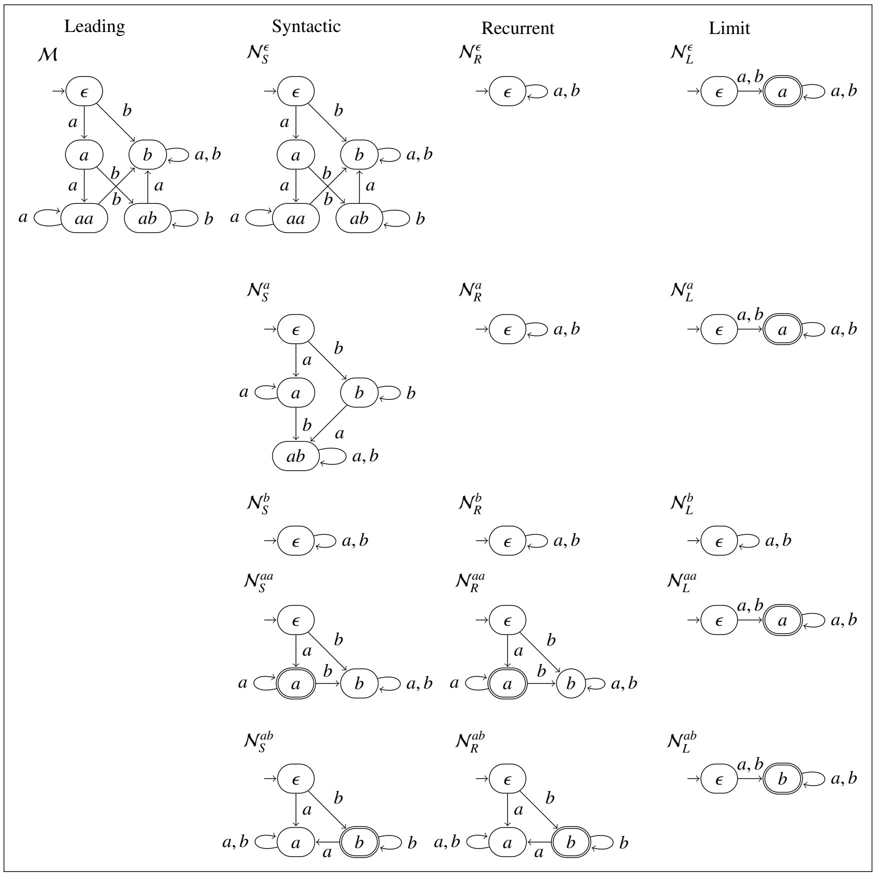

Fig. 1. Three types of FDFAs for $L = a^\omega + ab^\omega$. The final states are marked with double lines.

2. As an immediate result of the Item (1), we have that $| \approx^u_L | \leq | \approx^u_S | \leq | \backsim | \cdot | \approx^u_L |$. We prove the second claim by showing that, for all $u, x, y \in \Sigma^*$, if $ux \backsim uy$ and $x \approx^u_P y$, then $x \approx^u_S y$ (and thus $x \approx^u_L y$). Fix a word $v \in \Sigma^*$. Since $ux \backsim uy$ holds, it follows that $ux \cdot v \backsim u \iff uy \cdot v \backsim u$. Moreover, we have $u \cdot (xv)^\omega \in L \iff u \cdot (yv)^\omega \in L$ because $x \approx^u_P y$ holds. By definition of $\approx^u_S$, it follows that $x \approx^u_S y$ holds. Hence, $x \approx^u_L y$ holds as well. We then conclude that $| \approx^u_L | \leq | \backsim | \cdot | \approx^u_P |$. □

According to Definition 1, we have $x \backsim y$ iff $\mathcal{T}[\backsim](x) = \mathcal{T}[\backsim](y)$ for all $x, y \in \Sigma^*$. That is, $\mathcal{M} = (\mathcal{T}[\backsim], \emptyset)$ is consistent with $\backsim$, i.e., $x \backsim y$ iff $\mathcal{M}(x) = \mathcal{M}(y)$ for all $x, y \in \Sigma^*$. Hence, $u \cdot v \backsim u$ iff $\mathcal{M}(u) = \mathcal{M}(u \cdot v)$. In the remaining part of the paper, we may therefore mix the use of $\backsim$ and $\mathcal{M}$ without distinguishing the two notations.

We are now ready to give our main result of this section.

Theorem 2. *Let L be an ω-regular language and $\mathcal{F}_L$=($\mathcal{M}[\backsim]$, $\{\mathcal{N}[\approx_u]\}_{[u]_\backsim \in \Sigma^*/_\backsim}$) be the limit FDFA of L. Then (1) $\mathcal{F}_L$ has a finite number of states, (2) $UP(\mathcal{F}_L) = UP(L)$, and (3) $\mathcal{F}_L$ is saturated.*

Proof. Since the syntactic FDFA $\mathcal{F}_S$ of L has a finite number of states [17] and $\mathcal{F}_L$ is a coarser representation than $\mathcal{F}_S$ (cf. Lemma 1), $\mathcal{F}_L$ must have finite number of states as well.

To show $\mathrm{UP}(\mathcal{F}_L) \subseteq \mathrm{UP}(L)$, assume that $w \in \mathrm{UP}(\mathcal{F}_L)$. By Definition 3, a UP-word w is accepted by $\mathcal{F}_L$ if there exists a decomposition (u, v) of w such that $\mathcal{M}(u) = \mathcal{M}(u{\cdot}v)$ (equivalently, $u{\cdot}v \backsim u$) and $v \in \mathcal{L}_*(\mathcal{N}_L^{\tilde{u}})$ where $\tilde{u} = \mathcal{M}(u)$. Here $\tilde{u}$ is the representative word for the equivalence class $[u]_\backsim$. Similarly, let $\tilde{v} = \mathcal{N}_L^{\tilde{u}}(v)$. By Definition 9, we have $\tilde{u} \cdot \tilde{v} \backsim \tilde{u} \implies \tilde{u} \cdot \tilde{v}^\omega \in L$ holds as $\tilde{v}$ is a final state of $\mathcal{N}_L^{\tilde{u}}$. Since $v \approx_L^{\tilde{u}} \tilde{v}$ (i.e., $\mathcal{N}_L^{\tilde{u}}(v) = \mathcal{N}_L^{\tilde{u}}(\tilde{v})$), $\tilde{u} \cdot v \backsim \tilde{u} \implies \tilde{u} \cdot v^\omega \in L$ holds as well. It follows that $u \cdot v \backsim u \implies u \cdot v^\omega \in L$ since $u \backsim \tilde{u}$ and $u \cdot v \backsim \tilde{u} \cdot v$ (equivalently, $\mathcal{M}(u \cdot v) = \mathcal{M}(\tilde{u} \cdot v)$). Together with the assumption that $\mathcal{M}(u \cdot v) = \mathcal{M}(u)$ (i.e, $u \backsim u \cdot v$), we then have that $u \cdot v^\omega \in L$ holds. So, $\mathrm{UP}(\mathcal{F}_L) \subseteq \mathrm{UP}(L)$ also holds.

To show that $\mathrm{UP}(L) \subseteq \mathrm{UP}(\mathcal{F}_L)$ holds, let $w \in \mathrm{UP}(L)$. For a UP-word $w \in L$, we can find a normalized decomposition (u, v) of w such that $w = u \cdot v^\omega$ and $u \cdot v \backsim u$ (i.e., $\mathcal{M}(u) = \mathcal{M}(u \cdot v)$), since the index of $\backsim$ is finite (cf. [3] for more details). Let $\tilde{u} = \mathcal{M}(u)$ and $\tilde{v} = \mathcal{N}_L^{\tilde{u}}(v)$. Our goal is to prove that $\tilde{v}$ is a final state of $\mathcal{N}_L^{\tilde{u}}$. Since $u \backsim \tilde{u}$ and $u{\cdot}v^\omega \in L$, then $\tilde{u}{\cdot}v^\omega \in L$ holds. Moreover, $\tilde{u}{\cdot}v \backsim \tilde{u}$ holds as well because $\tilde{u} = \mathcal{M}(\tilde{u}) = \mathcal{M}(u) = \mathcal{M}(\tilde{u}{\cdot}v) = \mathcal{M}(u{\cdot}v)$. (Recall that $\mathcal{M}$ is deterministic.) Hence, $\tilde{u} \cdot v \backsim \tilde{u} \implies \tilde{u} \cdot v^\omega \in L$ holds. Since $\tilde{v} \approx_L^{\tilde{u}} v$, it follows that $\tilde{u} \cdot \tilde{v} \backsim \tilde{u} \implies \tilde{u} \cdot \tilde{v}^\omega \in L$ also holds. Hence, $\tilde{v}$ is a final state. Therefore, (u, v) is accepted by $\mathcal{F}_L$, i.e., $w \in \mathrm{UP}(\mathcal{F}_L)$. It follows that $\mathrm{UP}(L) \subseteq \mathrm{UP}(\mathcal{F}_L)$.

Now we show that $\mathcal{F}_L$ is saturated. Let w be a UP-word. Let (u, v) and (x, y) be two normalized decompositions of w with respect to $\mathcal{M}$ (or, equivalently, to $\backsim$). Assume that (u, v) is accepted by $\mathcal{F}_L$. From the proof above, it follows that both $u \cdot v \backsim u$ and $u \cdot v^\omega \in L$ hold. So, we know that $u \cdot v^\omega = x \cdot y^\omega \in L$. Let $\tilde{x} = \mathcal{M}(x)$ and $\tilde{y} = \mathcal{N}_L^{\tilde{x}}(y)$. Since (x, y) is a normalized decomposition, it follows that $x \cdot y \backsim x$. Again, since $\tilde{x} \backsim x$, $\tilde{x} \cdot y \backsim \tilde{x}$ and $\tilde{x} \cdot y^\omega \in L$ also hold. Obviously, $\tilde{x} \cdot y \backsim \tilde{x} \implies \tilde{x} \cdot y^\omega \in L$ holds. By the fact that $y \approx_L^{\tilde{x}} \tilde{y}$, $\tilde{x} \cdot \tilde{y} \backsim \tilde{x} \implies \tilde{x} \cdot \tilde{y}^\omega \in L$ holds as well. Hence, $\tilde{y}$ is a final state of $\mathcal{N}_L^{\tilde{x}}$. In other words, (x, y) is also accepted by $\mathcal{F}_L$. The proof for the case when (u, v) is not accepted by $\mathcal{F}_L$ is similar. □

3.2 Size comparison with other canonical FDFAs

As aforementioned, Angluin and Fisman in [3] showed that for a variant of the family of languages L_n given by Michel [18], its periodic FDFA has $\Omega(n!)$ states, while the syntactic FDFA only has $\mathcal{O}(n^2)$ states. Since limit FDFAs are smaller than syntactic FDFAs, it immediately follows that:

Corollary 2. *There exists a family of languages L_n such that its periodic FDFA has $\Omega(n!)$ states, while the limit FDFA only has $\mathcal{O}(n^2)$ states.*

Now we consider the size comparison between limit and recurrent FDFAs. Consider again the limit and recurrent FDFAs of the language $L = a^\omega + ab^\omega$ in Figure 1: one can see that limit FDFA and recurrent FDFA have the same number of states, even though with different progress DFAs. In fact, it is easy to see that limit FDFAs and recurrent FDFAs are incomparable regarding the their number of states, even when only the ω-regular languages recognized by weak DBAs are considered. A *weak* DBA (wDBA) is a DBA in which each SCC contains either all accepting transitions or non-accepting transitions.

Lemma 2. *If L is a wDBA-recognizable language, then its limit FDFA and its recurrent FDFA have incomparable size.*

Proof. We fix $u, x, y \in \Sigma^*$ in the proof. Since L is recognized by a wDBA, the TS $\mathcal{T}[\backsim]$ of the leading DFA $\mathcal{M}$ is isomorphic to the minimal wDBA recognizing L [16]. Therefore, a state $[u]_\backsim$ of $\mathcal{M}$ is either transient, in a rejecting SCC, or in an accepting SCC. We consider these three cases.

- Assume that $[u]_\backsim$ is a transient SCC/state. Then for all $v \in \Sigma^*$, $u \cdot x \cdot v \not\backsim u$ and $u \cdot y \cdot v \not\backsim u$.
  By the definitions of $\approx^u_R$ and $\approx^u_L$, there are a non-final class $[\epsilon]_{\approx^u_L}$ and *possibly* a sink final class $[\sigma]_{\approx^u_L}$ for $\approx^u_L$ where $\sigma \in \Sigma$, while there is a non-final class $[\epsilon]_{\approx^u_R}$ for $\approx^u_R$. Hence, $x \approx^u_L y$ implies $x \approx^u_R y$.
- Assume that $[u]_\backsim$ is in a rejecting SCC. Obviously, for all $v \in \Sigma^*$, we have that $u \cdot x \cdot v \backsim u \implies u \cdot (x \cdot v)^\omega \notin L$ and $u \cdot y \cdot v \backsim u \implies u \cdot (y \cdot v)^\omega \notin L$. Therefore, there is only one equivalence class $[\epsilon]_{\approx^u_R}$ for $\approx^u_R$. It follows that $x \approx^u_L y$ implies $x \approx^u_R y$.
- Assume that $[u]_\backsim$ is in an accepting SCC. Clearly, for all $v \in \Sigma^*$, we have that both $u \cdot x \cdot v \backsim u \implies u \cdot (x \cdot v)^\omega \in L$ and $u \cdot y \cdot v \backsim u \implies u \cdot (y \cdot v)^\omega \in L$ hold. That is, we have either $u \cdot x \cdot v \backsim u \wedge u \cdot (x \cdot v)^\omega \in L$ hold, or $u \cdot x \cdot v \not\backsim u$. If $x \approx^u_R y$ holds, it immediately follows that $(u \cdot x \cdot v \backsim u \implies u \cdot (x \cdot v)^\omega \in L) \iff (u \cdot y \cdot v \backsim u \implies u \cdot (y \cdot v)^\omega \in L)$ holds. Hence, $x \approx^u_R y$ implies $x \approx^u_L y$.

Based on this argument, it is easy to find a language L such that its limit FDFA is more succinct than its recurrent FDFA and vice versa, depending on the size comparison between rejecting SCCs and accepting SCCs. Therefore, the lemma follows. □

Lemma 2 reveals that limit FDFAs and recurrent FDFAs are incomparable in size. Nonetheless, we still provide a family of languages L_n in Lemma 3 such that the recurrent FDFA has $\Theta(n^2)$ states, while its limit FDFA only has $\Theta(n)$ states. One can, of course, obtain the opposite result by complementing L_n. Notably, Lemma 3 also gives a matching lower bound for the size comparison between syntactic FDFAs and limit FDFAs, since syntactic FDFAs can be quadratically larger than their limit FDFA counterparts, as stated in Lemma 1.

Lemma 3. *Let $\Sigma_n = \{0, 1, \cdots, n\}$. There exists an ω-regular language L_n over Σ_n such that its limit FDFA has $\Theta(n)$ states, while both its syntactic and recurrent FDFAs have $\Theta(n^2)$ states.*

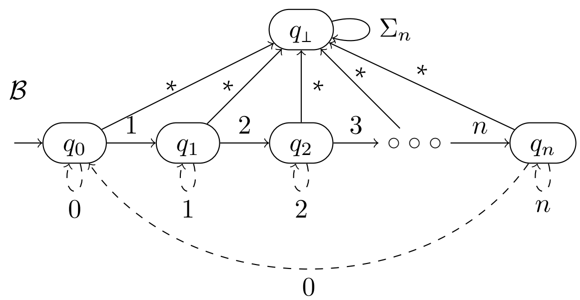

Fig. 2. The ω-regular language L_n represented with a DBA $\mathcal{B}$. The dashed arrows are Γ-transitions and $*$-transitions represent the missing transitions.

Proof. The family of languages L_n is defined as the language of the DBA $\mathcal{B} = (Q, \Sigma_n, q_0, \delta, \Gamma)$ as shown in Figure 2, where $\Sigma_n = \{0, 1, \cdots, n\}$. First, one can easily verify that the index of $\backsim_{L_n}$ is $n+2$. Here we add the subscript L_n to $\backsim_{L_n}$ to distinguish it from $\backsim$ for the language L we fix for the whole paper. In fact, the leading DFA induced by $\backsim_{L_n}$ is the exactly the TS of $\mathcal{B}$. Here, we only show that the limit FDFA and the recurrent FDFA of L_n, have $\Theta(n)$ states and $\Theta(n^2)$ states, respectively. We refer to Appendix A for detailed proofs of this lemma.

Now we fix a word u and consider the index of $\approx^u_L$. Let $x \in \Sigma^*$. Obviously, if $q_\perp = \mathcal{B}(u)$, then for all $v \in \Sigma^*$, we have $u \cdot x \cdot v \backsim_{L_n} u$ but $u \cdot (x \cdot v)^\omega \notin L_n$. Hence, $|\approx^u_L| = 1$. Now let $q_i = \mathcal{B}(u)$ with $0 \leq i \leq n$. For all $v \in \Sigma^*$, if $u \cdot x \cdot v \backsim_{L_n} u$ holds, it must be the case that $u \cdot (x \cdot v)^\omega \in L_n$ unless $x \cdot v = \epsilon$. Hence, $|\approx^u_L| = 2$. It follows that the limit FDFA of L_n has exactly $2 \times (n+1) + 1 + n + 2 \in \Theta(n)$ states.

Now we consider the index of $\approx^u_R$ for a fixed $u \in \Sigma^*$. Similarly, when $q_\perp = \mathcal{B}(u)$, $|\approx^u_R| = 1$ since for all $v \in \Sigma^*$, we have $u \cdot x \cdot v \backsim_{L_n} u \wedge u \cdot (x \cdot v)^\omega \notin L_n$. Now we consider that $q_k = \mathcal{B}(u)$ with $0 \leq k \leq n$. Let $x_1, x_2 \in \Sigma^*$. First, assume that $\mathcal{B}(u \cdot x_1) \neq \mathcal{B}(u \cdot x_2)$. W.l.o.g., let $q_j = \mathcal{B}(u \cdot x_2)$ with $0 \leq j \leq n$ and let $q_i = \mathcal{B}(u \cdot x_1)$ with either $i < j$ or $q_i = q_\perp$. We can easily construct a finite word v such that $q_k = \mathcal{B}(u) = \mathcal{B}(u \cdot x_2 \cdot v)$, i.e., $u \cdot x_2 \cdot v \backsim_{L_n} u$, and $u \cdot (x_2 \cdot v)^\omega \in L_n$. For example, we can let $v = (j+1) \cdots n \cdot 0 \cdots k$ if $j < k \leq n$. Hence, $u \cdot x_2 \cdot v \backsim_{L_n} u \wedge u \cdot (x_2 \cdot v)^\omega \in L_n$ holds. On the contrary, it is easy to see that $q_\perp = \mathcal{B}(u \cdot x_1 \cdot v) = \delta(q_i, j+1)$ since either $j+1 > i+1$ or $q_i = q_\perp$. In other words, we have $u \cdot x_1 \cdot v \not\backsim_{L_n} u \wedge u \cdot (x_1 \cdot v)^\omega \notin L_n$. By definition of $\approx^u_R$, $x_1 \not\approx^u_R x_2$. Hence, $|\approx^u_R| \geq n+2$. Next, we assume that $\mathcal{B}(u \cdot x_1) = \mathcal{B}(u \cdot x_2)$. For a word $v \in \Sigma^*$, it is easy to see that $u \cdot x_1 \cdot v \backsim_{L_n} u \Longleftrightarrow u \cdot x_2 \cdot v \backsim_{L_n} u$. Moreover, since $u \cdot x_1 \cdot v \backsim_{L_n} u$ implies $u \cdot (x_1 \cdot v)^\omega \in L_n$, we thus have that $u \cdot x_1 \cdot v \backsim_{L_n} u \wedge u \cdot (x_1 \cdot v)^\omega \in L_n \Longleftrightarrow u \cdot x_2 \cdot v \backsim_{L_n} u \wedge u \cdot (x_2 \cdot v)^\omega \in L_n$. In other words, $x_1 \approx^u_R x_2$, which implies that $|\approx^u_R| \leq n+2$. Hence $|\approx^u_R| = n+2$ when $\mathcal{B}(u) \neq q_\perp$. It follows that the recurrent FDFA of L_n has exactly $(n+2) \times (n+1) + 1 + (n+2) \in \Theta(n^2)$ states. □

Finally, it is time to derive yet another "Myhill-Nerode" theorem for ω-regular languages, as stated in Theorem 3. This result follows immediately from Lemma 1 and a similar theorem about syntactic FDFAs [17].

Theorem 3. *Let $\mathcal{F}_L$ be the limit FDFA of an ω-language L. Then L is regular if, and only if $\mathcal{F}_L$ has finite number of states.*

For identifying whether L is DBA-recognizable with FDFAs, a straight forward way as mentioned in the introduction is to go through determinization, which is, however, exponential in the size of the input FDFA. We show in Section 4 that there is a polynomial-time algorithm using our limit FDFAs.

4 Limit FDFAs for identifying DBA-recognizable languages

Given an ω-regular language L, we show in this section how to use the limit FDFA of L to check whether L is DBA-recognizable in polynomial time. To this end, we will first introduce how the limit FDFA of L looks like in Section 4.1 and then introduce the deciding algorithm in Section 4.2.

4.1 Limit FDFA for DBA-recognizable languages

Bohn and Löding [5] construct a type of family of DFAs $\mathcal{F}_{BL}$ from a set S^+ of positive samples and a set S^- of negative samples, where the progress DFA accepts exactly the language $V_u = \{x \in \Sigma^+ : \forall v \in \Sigma^*.$ if $u \cdot xv \backsim u$, then $u \cdot (xv)^\omega \in L\}$[4]. When the samples S^+ and S^- uniquely characterize a DBA-recognizable language L, $\mathcal{F}_{BL}$ recognizes exactly L.

The progress DFA $\mathcal{N}^u_L$ of our limit FDFA $\mathcal{F}_L$ of L usually accepts *more* words than V_u. Nonetheless, we can still find one final equivalence class that is exactly the set V_u, as stated in Lemma 4.

Lemma 4. *Let L be a DBA-recognizable language and $\mathcal{F}_L = (\mathcal{M}, \{\mathcal{N}^u_L\}_{[u]_\backsim \in \Sigma^*/_\backsim})$ be the limit FDFA of L. Then, for each progress DFA $\mathcal{N}^u_L$ with $\mathcal{L}_*(\mathcal{N}^u_L) \neq \emptyset$, there must exist a final state $\tilde{x} \in F_u$ such that $[\tilde{x}]_{\approx^u_L} = \{x \in \Sigma^+ : \forall v \in \Sigma^*.\ u \cdot (x \cdot v) \backsim u \implies u \cdot (x \cdot v)^\omega \in L\}$.*

Proof. In [5], it is shown that for each equivalence class $[u]_\backsim$ of $\backsim$, there exists a regular language $V_u = \{x \in \Sigma^+ : \forall v \in \Sigma^*.$ if $u \cdot xv \backsim u$, then $u \cdot (xv)^\omega \in L\}$. We have also provided the proof of the existence of V_u in Appendix C, adapted to our notations. The intuition of V_u is the following. Let $\mathcal{B} = (\Sigma, Q, \iota, \delta, \Gamma)$ be a DBA accepting L. Then, $[u]_\backsim$ corresponds to a set of states $S = \{q \in Q : q = \delta(\iota, u'), u' \in [u]_\backsim\}$ in $\mathcal{B}$. For each $q \in S$, we can easily create a regular language V_q such that $x \in V_q$ iff over the word x, $\mathcal{B}^q$ (the DBA derived from $\mathcal{B}$ by setting q its initial state) visits an accepting transition, $\mathcal{B}^q$ goes to an SCC that cannot

[4] Defining directly a progress RC $\approx^u$ that recognizes V_u is hard since V_u is quantified over all v-extensions.

go back to q, or $\mathcal{B}^q$ goes to a state that cannot go back to q unless visiting an accepting transition. Then, $V_u = \cap_{q \in S} V_q$.

Now we show that V_u is an equivalence class of $\approx^u_L$ as follows. On one hand, for every two different words $x_1, x_2 \in V_u$, we have that $x_1 \approx^u_L x_2$, which is obvious by the definition of V_u. On the other hand, it is easy to see that $x' \not\approx^u_L x$ for all $x' \notin V_u$ and $x \in V_u$ because there exists some $v \in \Sigma^*$ such that $u \cdot x' \cdot v \backsim u$ but $u \cdot (x' \cdot v)^\omega \notin L$. Hence, V_u is indeed an equivalence class of $\approx^u_L$. Obviously, $V_u \subseteq \mathcal{L}_*(\mathcal{N}^u_L)$, as we can let $v = \epsilon$, so for every word $x \in V_u$, we have that $u \cdot x \backsim u \implies u \cdot x^\omega \in L$. Let $\tilde{x} = \mathcal{N}^u_L(x)$ for a word $x \in V_u$. It follows that $\tilde{x}$ is a final state of $\mathcal{N}^u_L$ and we have $[\tilde{x}]_{\approx^u_L} = V_u$. This completes the proof. □

By Lemma 4, we can define a variant of limit FDFAs for only DBAs with less number of final states. This helps to reduce the complexity when translating FDFAs to NBAs [2,7,13]. Let n be the number of states in the leading DFA $\mathcal{M}$ and k be the number of states in the largest progress DFA. Then the resultant NBA from an FDFA has $\mathcal{O}(n^2k^3)$ states [2,7,13]. However, if the input FDFA is $\mathcal{F}_B$ as in Definition 10, the complexity of the translation will be $\mathcal{O}(n^2k^2)$, as there is at most one final state, rather than k final states, in each progress DFA.

Definition 10 (Limit FDFAs for DBAs). *The limit FDFA* $\mathcal{F}_B = (\mathcal{M}, \{\mathcal{N}^u_B\})$ *of* L *is defined as follows.*

The transition systems of $\mathcal{M}$ *and* $\mathcal{N}^u_B$ *for each* $[u]_\backsim \in \Sigma^*/_\backsim$ *are exactly the same as in Definition 9.*

The set of final states F_u *contains the equivalence classes* $[x]_{\approx^u_L}$ *such that, for all* $v \in \Sigma^*$, $u \cdot xv \backsim u \Longrightarrow u \cdot (xv)^\omega \in L$ *holds.*

The change to the definition of final states would not affect the language that the limit FDFAs recognize, but only their saturation properties. We say an FDFA $\mathcal{F}$ is *almost saturated* if, for all $u, v \in \Sigma^*$, we have that if (u, v) is accepted by $\mathcal{F}$, then (u, v^k) is accepted by $\mathcal{F}$ for all $k \geq 1$. According to [13], if $\mathcal{F}$ is almost saturated, then the translation algorithm from FDFAs to NBAs in [2,7,13] still applies (cf. Appendix B about details of the NBA construction).

Theorem 4. *Let* L *be a DBA-recognizable language and* $\mathcal{F}_B$ *be the limit FDFA induced by Definition 10. Then (1)* $UP(\mathcal{F}_B) = UP(L)$ *and (2)* $\mathcal{F}_B$ *is almost saturated but not necessarily saturated.*

Proof. The proof for $\mathrm{UP}(\mathcal{F}_B) \subseteq \mathrm{UP}(L)$ is trivial, as the final states defined in Definition 10 must also be final in Definition 9. The other direction can be proved based on Lemma 4. Let $w \in \mathrm{UP}(L)$ and $\mathcal{B} = (Q, \Sigma, \iota, \delta, \Gamma)$ be a DBA accepting L. Let ρ be the run of $\mathcal{B}$ over w. We can find a decomposition (u, v) of w such that there exists a state q with $q = \delta(\iota, u) = \delta(\iota, u \cdot v)$ and $(q, v[0]) \in \Gamma$. As in the proof of Lemma 4, we are able to construct the regular language $V_u = \{x \in \Sigma^+ : \forall y \in \Sigma^*, u \cdot x \cdot y \backsim u \implies u \cdot (x \cdot y)^\omega \in L\}$. We let $S = \{p \in Q : \mathcal{L}(\mathcal{B}^q) = \mathcal{L}(\mathcal{B}^p)\}$. For every state $p \in S$, we have that $v^\omega \in \mathcal{L}(\mathcal{B}^p)$. For each $p \in S$, we select an integer $k_p > 0$ such that the finite run $p \xrightarrow{v^{k_p}} \delta(p, v^{k_p})$ visits some accepting transition. Then we let $k = \max_{p \in S} k_p$. By definition of V_u, it follows that $v^k \in V_u$. That is, V_u is not empty. According to Lemma 4, we have a final equivalence class $[x]_{\approx^u_L} = V_u$ with $v^k \in [x]_{\approx^u_L}$. Moreover, $u \cdot v^k \backsim u$ since $q = \delta(\iota, u) = \delta(q, v)$. Hence, (u, v^k) is accepted by $\mathcal{F}_B$, i.e., $w \in \mathrm{UP}(\mathcal{F}_B)$. It follows that $\mathrm{UP}(\mathcal{F}_B) = \mathrm{UP}(L)$.

Now we prove that $\mathcal{F}_B = (\mathcal{M}, \{\mathcal{N}^u_B\})$ is *not* necessarily saturated. Let $L = (\Sigma^* \cdot aa)^\omega$. Obviously, L is DBA recognizable, and $\backsim$ has only one equivalence class, $[\epsilon]_\backsim$. Let $w = a^\omega \in \mathrm{UP}(L)$. Let $(u = \epsilon, v = a)$ be a normalized decomposition of w with respect to $\backsim$ (thus, $\mathcal{M}$). We can see that there exists a finite word x (e.g., $x = b$ is such a word) such that $\epsilon \cdot a \cdot x \backsim \epsilon$ and $\epsilon \cdot (a \cdot x)^\omega \notin L$. Thus, (ϵ, a) will not be accepted by $\mathcal{F}_B$. Hence $\mathcal{F}_B$ is not saturated. Nonetheless, it is easy to verify that $\mathcal{F}_B$ is almost saturated. Assume that (u, v) is accepted by $\mathcal{F}_B$. Let $\tilde{u} = \mathcal{M}(u)$ and $\tilde{v} = \mathcal{N}^{\tilde{u}}_B(v)$. Since $\tilde{v}$ is the final state, then, according to Definition 10, we have for all $e \in \Sigma^*$ that $\tilde{u} \cdot \tilde{v}e \backsim \tilde{u} \implies \tilde{u} \cdot (\tilde{v}e)^\omega \in L$. Since $v \approx^u_L \tilde{v}$, $\tilde{u} \cdot ve \backsim \tilde{u} \implies \tilde{u} \cdot (ve)^\omega \in L$ also holds for all $e \in \Sigma^*$. Let $e = v^k \cdot e'$ where $e' \in \Sigma^*, k \geq 0$. It follows that $\tilde{u} \cdot v^k e' \backsim \tilde{u} \implies \tilde{u} \cdot (v^k e')^\omega \in L$ holds for $k \geq 1$ as well. Therefore, for all $e' \in \Sigma^*, k \geq 1$, $(\tilde{u} \cdot \tilde{v}e' \backsim \tilde{u} \implies \tilde{u} \cdot (\tilde{v}e')^\omega \in L) \iff (\tilde{u} \cdot v^k e' \backsim \tilde{u} \implies \tilde{u} \cdot (v^k e')^\omega \in L)$ holds. In other words, $\tilde{v} \approx^{\tilde{u}}_L v^k$ for all $k \geq 1$. Together with that $uv^k \backsim u$, (u, v^k) is accepted by $\mathcal{F}_B$ for all $k \geq 1$. Hence, $\mathcal{F}_B$ is almost saturated. □

4.2 Deciding DBA-recognizable languages

We show next how to identify whether a language L is DBA-recognizable with our limit FDFA $\mathcal{F}_L$. Our decision procedure relies on the translation of FDFAs to NBAs/DBAs. In the following, we let n be the number of states in the leading DFA $\mathcal{M}$ and k be the number of states in the largest progress DFA. We first give some previous results below.

Lemma 5 ([13, Lemma 6]). *Let $\mathcal{F}$ be an (almost) saturated FDFA of L. Then one can construct an NBA $\mathcal{A}$ with $\mathcal{O}(n^2 k^3)$ states such that $\mathcal{L}(\mathcal{A}) = L$.*

Now we consider the translation from FDFA to DBAs. By Lemma 4, there is a final equivalence class $[x]_{\approx^u_L}$ that is a *co-safety* language in the limit FDFA of L. Co-safety regular languages are regular languages $R \subseteq \Sigma^*$ such that $R \cdot \Sigma^* = R$. It is easy to verify that if $x' \in [x]_{\approx^u_L}$, then $x'v \in [x]_{\approx^u_L}$ for all $v \in \Sigma^*$, based on the definition of $\approx^u_L$. So, $[x]_{\approx^u_L}$ is a co-safety language. The DFAs accepting co-safety languages usually have a sink final state f (such that f transitions to itself over all letters in Σ). We therefore have the following.

Corollary 3. *If L is DBA-recognizable then every progress DFA $\mathcal{N}^u_L$ of the limit FDFA $\mathcal{F}_L$ of L either has a sink final state, or no final state at all.*

Our limit FDFA $\mathcal{F}_B$ of L, as constructed in Definition 10, accepts the same co-safety languages in the progress DFAs as the FDFA obtained in [5], although they may have different transition systems. Nonetheless, we show that their DBA construction still works on $\mathcal{F}_B$. To make the construction more general, we assume an FDFA $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\}_{q \in Q})$ where $\mathcal{M} = (Q, \Sigma, \iota, \delta)$ and, for each $q \in Q$, we have $\mathcal{N}^q = (Q_q, \Sigma, \iota_q, \delta_q, F_q)$.

Definition 11 ([5]). *Let $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\}_{q \in Q})$ be an FDFA. Let $\mathcal{T}[\mathcal{F}]$ be the TS constructed from $\mathcal{F}$ defined as the tuple $\mathcal{T}[\mathcal{F}] = (Q_{\mathcal{T}}, \Sigma, \iota_{\mathcal{T}}, \delta_{\mathcal{T}})$ and $\Gamma \subseteq \{(q, \sigma) : q \in Q_{\mathcal{T}}, \sigma \in \Sigma\}$ be a set of transitions where*

- $Q_{\mathcal{T}} := Q \times \bigcup_{q \in Q} Q_q$;
- $\iota_{\mathcal{T}} := (\iota, \iota_\iota)$;
- *For a state $(m, q) \in Q_{\mathcal{T}}$ and $\sigma \in \Sigma$, let $q' = \delta_{\widetilde{m}}(q, \sigma)$ where $\mathcal{N}^{\widetilde{m}}$ is the progress DFA that q belongs to and let $m' = \delta(m, \sigma)$. Then*

$$\delta((m,q),\sigma) = \begin{cases} (m', q') & \text{if } q' \notin F_{\widetilde{m}} \\ (m', \iota_{m'}) & \text{if } q' \in F_{\widetilde{m}} \end{cases}$$

- $((m, q), \sigma) \in \Gamma$ *if* $q' \in F_{\widetilde{m}}$

Lemma 6. *If $\mathcal{F}$ is an FDFA with only sink final states. Let $\mathcal{B}[\mathcal{F}] = (\mathcal{T}[\mathcal{F}], \Gamma)$ as given in Definition 11. Then, UP$(\mathcal{L}(\mathcal{B}[\mathcal{F}])) \subseteq$ UP$(\mathcal{F})$.*

Proof. Let $w \in \text{UP}(\mathcal{L}(\mathcal{B}[\mathcal{F}]))$ and ρ be its corresponding accepting run. Since w is a UP-word and $\mathcal{B}[\mathcal{F}]$ is a DBA of finite states, then we must be able to find a decomposition (u, v) of w such that $(m, \iota_m) = \mathcal{B}[\mathcal{F}](u) = \mathcal{B}[\mathcal{F}](u \cdot v)$, where ρ will visit a Γ-transition whose destination is (m, ι_m) for infinitely many times. It is easy to see that $\mathcal{M}(u \cdot v) = \mathcal{M}(u)$ since $\mathcal{B}[\mathcal{F}](u) = \mathcal{B}[\mathcal{F}](u \cdot v)$. Moreover, we can show there must be a prefix of v, say v', such that $v' \in \mathcal{L}_*(\mathcal{N}^m)$. Since $\mathcal{L}_*(\mathcal{N}^m)$ is co-safety, we have that $v \in \mathcal{L}_*(\mathcal{N}^m)$. Thus, (u, v) is accepted by $\mathcal{F}$. By Definition 3, $w \in \text{UP}(\mathcal{F})$. Therefore, $\text{UP}(\mathcal{L}(\mathcal{B}[\mathcal{F}])) \subseteq \text{UP}(\mathcal{F})$. □

By Corollary 3, $\mathcal{F}_B$ has only sink final states; so, we have that $\text{UP}(\mathcal{L}(\mathcal{B}[\mathcal{F}_B])) \subseteq \text{UP}(\mathcal{F}_B)$. However, Corollary 3 is only a necessary condition for L being DBA-recognizable, as explained below. Let L be an ω-regular language over $\Sigma = \{1, 2, 3, 4\}$ such that a word $w \in L$ iff the maximal number that occurs infinitely often in w is even. Clearly, L has one equivalence class $[\epsilon]_{\backsim}$ for $\backsim$. The limit FDFA $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}_L^\epsilon\})$ of L is depicted in Figure 3. We can observe

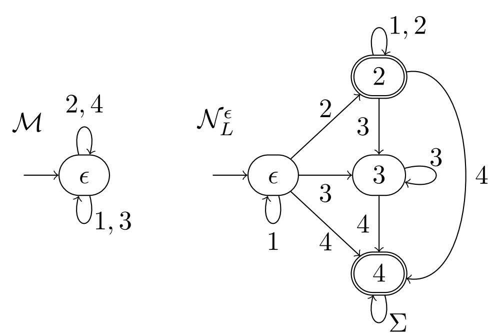

Fig. 3. An example limit FDFA $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}_L^\epsilon\})$

that the equivalence class $[4]_{\approx^{\epsilon}_{L}}$ corresponds to a co-safety language. Hence, the progress DFA $\mathcal{N}^{\epsilon}_{L}$ has a sink final state. However, L is not DBA-recognizable. If we ignore the final equivalence class $[2]_{\approx^{\epsilon}_{L}}$ and obtain the variant limit FDFA $\mathcal{F}_B$ as given in Definition 10, then we have $\mathrm{UP}(\mathcal{F}_B) \neq \mathrm{UP}(L)$ since the ω-word 2^{ω} is missing. But then, by Theorem 4, this change would not lose words in L if L is DBA-recognisable, leading to contradiction. Therefore, L is shown to be not DBA-recognizable. So the key of the decision algorithm here is to check whether ignoring other final states will retain the language. With Lemma 7, we guarantee that $\mathcal{B}[\mathcal{F}_B]$ accepts exactly L if L is DBA-recognizable.

Lemma 7. *Let L be a DBA-recognizable language. Let $\mathcal{F}_B$ be the limit FDFA L, as constructed in Definition 10. Let $\mathcal{B}[\mathcal{F}_B] = (\mathcal{T}[\mathcal{F}_B], \Gamma)$, where $\mathcal{T}[\mathcal{F}_B]$ and Γ are the TS and set of transitions, respectively, defined in Definition 11 from $\mathcal{F}_B$. Then $UP(\mathcal{F}_B) = UP(L) \subseteq UP(\mathcal{L}(\mathcal{B}[\mathcal{F}_B]))$.*

Proof. We first assume for contradiction that some $w \in L$ is rejected by $\mathcal{B}[\mathcal{F}_B]$. For this, we consider the run $\rho = (q_0, w[0], q_1)(q_1, w[1], q_2)\ldots$ of $\mathcal{B}[\mathcal{F}_B]$ on w. Let $i \in \omega$ be such that $(q_{i-1}, w[i{-}1], q_i)$ is the last accepting transition in ρ, and $i = 0$ if there is no accepting transition at all in ρ. We also set $u = w[0 \cdots i-1]$ and $w' = w[i \cdots]$. By Definition 11, this ensures that $\mathcal{B}[\mathcal{F}_B]$ is in state $([u]_{\backsim}, \iota_{[u]_{\backsim}})$ after reading u and will not see accepting transitions (or leave $\mathcal{N}_B^{[u]_{\backsim}}$) while reading the tail w'.

Let $\mathcal{D} = (Q', \Sigma, \iota', \delta', \Gamma')$ be a DBA that recognizes L and has only reachable states. As $\mathcal{D}$ recognizes L, it has the same right congruences as L; by slight abuse of notation, we refer to the states in Q' that are language equivalent to the state reachable after reading u by $[u]_{\backsim}$ and note that $\mathcal{D}$ is in some state of $[u]_{\backsim}$ after (and only after) reading a word $u' \backsim u$.

As $u \cdot w'$, and therefore $u' \cdot w'$ for all $u' \backsim u$, are in L, they are accepted by $\mathcal{D}$, which in particular means that, for all $q \in [u]_{\backsim}$, there is an i_q such that there is an accepting transition in the first i_q steps of the run of $\mathcal{D}^q$ (the DBA obtained from $\mathcal{D}$ by setting the initial state to q) on w'. Let i_+ be maximal among them and $v = w[i \cdots i + i_+]$. Then, for $u' \backsim u$ and any word $u'vv'$, we either have $u'vv' \not\backsim u$, or $u'vv' \backsim u$ *and* $u' \cdot (vv')^{\omega} \in L$. (The latter is because v is constructed such that a run of $\mathcal{D}$ on this word will see an accepting transition while reading each v, and thus infinitely many times.) Thus, $\mathcal{N}_B^{[u]_{\backsim}}$ will accept any word that starts with v, and therefore be in a final sink after having read v.

But then $\mathcal{B}[\mathcal{F}_B]$ will see another accepting transition after reading v (at the latest after having read uv), which closes the contradiction and completes the proof. ☐

So, our decision algorithm works as follows. Assume that we are given the limit FDFA $\mathcal{F}_L = (\mathcal{M}, \{\mathcal{N}^q_L\})$ of L.

1. We first check whether there is a progress DFA $\mathcal{N}^q_L$ such that there are final states but without the sink final state. If it is the case, we terminate and return "NO".

2. Otherwise, we obtain the FDFA $\mathcal{F}_B$ by keeping the sink final state as the sole final state in each progress DFA (cf. Definition 10). Let $\mathcal{A} = \mathtt{NBA}(\mathcal{F}_L)$ be the NBA constructed from $\mathcal{F}_L$ (cf. Lemma 5) and $\mathcal{B} = \mathtt{DBA}(\mathcal{F}_B)$ be the DBA constructed from $\mathcal{F}_B$ (cf. Definition 11). Obviously, we have that $\mathrm{UP}(\mathcal{L}(\mathcal{A})) = \mathrm{UP}(L)$ and $\mathrm{UP}(\mathcal{L}(\mathcal{B})) \subseteq \mathrm{UP}(\mathcal{F}_B) = \mathrm{UP}(L)$.
3. Then we check whether $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$ holds. If so, we return "YES", and otherwise "NO".

Now we are ready to give the main result of this section.

Theorem 5. *Deciding whether L is DBA-recognizable can be done in time polynomial in the size of the limit FDFA of L.*

Proof. We first prove our decision algorithm is correct. If the algorithm returns "YES", clearly, we have $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B})$. It immediately follows that $\mathrm{UP}(L) = \mathrm{UP}(\mathcal{L}(\mathcal{A})) \subseteq \mathrm{UP}(\mathcal{L}(\mathcal{B})) \subseteq \mathrm{UP}(\mathcal{F}_B) \subseteq \mathrm{UP}(\mathcal{F}_L) = \mathrm{UP}(L)$ according to Lemmas 5 and 6. Hence, $\mathrm{UP}(\mathcal{L}(\mathcal{B})) = \mathrm{UP}(L)$, which implies that L is DBA-recognizable. For the case that the algorithm returns "NO", we analyze two cases:

1. $\mathcal{F}$ has final states but without sink accepting states for some progress DFA. By Corollary 3, L is not DBA-recognizable.
2. $\mathcal{L}(\mathcal{A}) \not\subseteq \mathcal{L}(\mathcal{B})$. It means that $\mathrm{UP}(L) \not\subseteq \mathrm{UP}(\mathcal{L}(\mathcal{B}))$ (by Lemma 5). It follows that L is not DBA-recognizable by Lemma 7.

The algorithm is therefore sound; its completeness follows from Lemmas 6 and 7.

The translations above are all in polynomial time. Moreover, checking the language inclusion between an NBA and a DBA can also be done in polynomial time [12]. Hence, the deciding algorithm is also in polynomial time in the size of the limit FDFA of L. □

Recall that, our limit FDFAs are dual to recurrent FDFAs. One can observe that, for DBA-recognizable languages, recurrent FDFAs do not necessarily have sink final states in progress DFAs. For instance, the ω-regular language $L = a^\omega + ab^\omega$ is DBA-recognizable, but its recurrent FDFA, depicted in Fig. 1, does not have sink final states. Hence, our deciding algorithm does not work with recurrent FDFAs.

5 Underspecifying progress right congruences

Recall that recurrent and limit progress DFAs $\mathcal{N}^u$ either treat don't care words in $\overline{C_u} = \{v \in \Sigma^+ : uv \not\backsim u\}$ as rejecting or accepting, whereas it really does not matter whether or not they are accepted. So why not keep this question open? We do just this in this section; however, we find that treating the progress with maximal flexibility comes at a cost: the resulting right progress relation $\approx^u_N$ is *no* longer an equivalence relation, but only a reflexive and symmetric relation over $\Sigma^* \times \Sigma^*$ such that $x \approx^u_N y$ implies $xv \approx^u_N yv$ for all $u, x, y, v \in \Sigma^*$.

For this, we first introduce *Right Pro-Congruences* (RP) as relations on words that satisfy all requirements of an RC except for transitivity.

Definition 12 (Progress RP). *Let $[u]_{\backsim}$ be an equivalence class of $\backsim$. For $x, y \in \Sigma^*$, we define the progress RP $\approx_N^u$ as follows:*

$$x \approx_N^u y \text{ iff } \forall v \in \Sigma^*.\ (uxv \backsim u \wedge uyv \backsim u) \implies (u \cdot (xv)^\omega \in L \Longleftrightarrow u \cdot (yv)^\omega \in L).$$

Obviously, $\approx_N^u$ is a RP, i.e., for $x, y, v' \in \Sigma^\omega$, if $x \approx_N^u y$, then $xv' \approx_N^u yv'$. That is, assume that $x \approx_N^u y$ and we want to prove that, for all $e \in \Sigma^*$, $(u \cdot xv'e \backsim u \wedge u \cdot yv'e \backsim u) \implies (u \cdot (xv'e)^\omega \in L \Longleftrightarrow u \cdot (yv'e)^\omega \in L)$. This follows immediately by setting $v = v'e$ in Definition 12 for all $e \in \Sigma^*$ since $x \approx_N^u y$. As $\approx_N^u$ is not necessarily an equivalence relation[5], so that we cannot argue directly with the size of its index. However, we can start with showing that $\approx_N^u$ is coarser than $\approx_P^u, \approx_S^u, \approx_R^u$, and $\approx_L^u$.

Lemma 8. *For $u, x, y \in \Sigma^*$, we have that if $x \approx_K^u y$, then $x \approx_N^u y$, where $K \in \{P, S, R, L\}$.*

Proof. First, if $x \approx_P^u y$, $x \approx_N^u y$ holds trivially.

For syntactic, recurrent, and limit RCs, we first argue for fixed $v \in \Sigma^*$ that

- $ux \backsim uy \Longrightarrow uxv \backsim uyv$, and therefore
  $ux \backsim uy \wedge \big(u \cdot x \cdot v \backsim u \Longrightarrow (u \cdot (x \cdot v)^\omega \in L \Longleftrightarrow u \cdot (y \cdot v)^\omega \in L)\big)$
  $\models (uxv \backsim u \wedge uyv \backsim u) \implies (u \cdot (xv)^\omega \in L \Longleftrightarrow u \cdot (yv)^\omega \in L)$,
- $(u \cdot x \cdot v \backsim u \wedge u \cdot (xv)^\omega \in L) \Longleftrightarrow (u \cdot yv \backsim u \wedge u \cdot (y \cdot v)^\omega \in L)$
  $\models (uxv \backsim u \wedge uyv \backsim u) \implies (u \cdot (xv)^\omega \in L \Longleftrightarrow u \cdot (yv)^\omega \in L)$, and
- $(u \cdot x \cdot v \backsim u \Longrightarrow u \cdot (x \cdot v)^\omega \in L) \Longleftrightarrow (u \cdot y \cdot v \backsim u \Longrightarrow u \cdot (y \cdot v)^\omega \in L)$
  $\models (uxv \backsim u \wedge uyv \backsim u) \implies (u \cdot (xv)^\omega \in L \Longleftrightarrow u \cdot (yv)^\omega \in L)$,

which is simple Boolean reasoning. As this holds for all $v \in \Sigma^*$ individually, it also holds for the intersection over all $v \in \Sigma^*$, so that the claim follows. □

Now, it is easy to see that we can use any RC $\approx$ that refines $\approx_N^u$ and use it to define a progress DFA. It therefore makes sense to define the set of RCs that refine $\approx_N^u$ as $\mathsf{RC}(\approx_N^u) = \{\approx \mid \approx \subset \approx_N^u \text{ is a RC}\}$, and the best index $|\approx_N^u|$ of our progress RP as $|\approx_N^u| = \min\{|\approx| \mid \approx \in \mathsf{RC}(\approx_N^u)\}$. With this definition, Corollary 4 follows immediately.

Corollary 4. *For $u \in \Sigma^*$, we have that $|\approx_N^u| \leq |\approx_K^u|$ for all $K \in \{P, S, R, L\}$.*

We note that the restriction of $\approx_N^u$ to $C_u \times C_u$ is still an equivalence relation, where $C_u = \{v \in \Sigma^* : uv \backsim u\}$ are the words the FDFA acceptance conditions really care about. This makes it easy to define a DFA over each $\approx \in \mathsf{RC}(\approx_N^u)$ with finite index: $C_u/_{\approx_N^u}$ is good if it contains a word v s.t. $u \cdot v^\omega \in L$, and a quotient of $\Sigma^*/_{\approx}$ is accepting if it intersects with a good quotient (note that it intersects with at most one quotient of C_u). With this preparation, we now show the following.

[5] In the language $L = a^\omega + ab^\omega$ from the example of Figure 1, for example, we have $a \approx_N^{ab} \epsilon$ and $a \approx_N^{ab} b$, but $b \not\approx_N^{ab} \epsilon$.

Theorem 6. *Let L be an ω-regular language and $\mathcal{F}_L{=}(\mathcal{M}[\backsim], \{\mathcal{N}[\approx_u]\}_{[u]_\backsim \in \Sigma^*/_\backsim})$ be the limit FDFA of L s.t. $\approx_u \in \mathsf{RC}(\approx_N^u)$ with finite index for all u. Then (1) $\mathcal{F}_L$ has a finite number of states, (2) $UP(\mathcal{F}_L) = UP(L)$, and (3) $\mathcal{F}_L$ is saturated.*

The proof is similar to the proof of Theorem 2 and moved to Appendix D.

6 Discussion and future work

Our limit FDFAs fit nicely into the learning framework for FDFAs [3] and are already available for use in the learning library ROLL[6] [14]. Since one can treat an FDFA learner as comprised of a family of DFA learners in which one DFA of the FDFA is learned by a separate DFA learner, we only need to adapt the learning procedure for progress DFAs based on our limit progress RCs, without extra development of the framework; see Appendix E for details. We leave the empirical evaluation of our limit FDFAs in learning ω-regular languages as future work.

We believe that limit FDFAs are complementing the existing set of canonical FDFAs, in terms of recognizing and learning ω-regular languages. Being able to easily identify DBA-recognizable languages, limit FDFAs might be used in a learning framework for DBAs using membership and equivalence queries. We leave this to future work. Finally, we have looked at retaining maximal flexibility in the construction of FDFA by moving from progress RCs to progress RPs. While this reduces size, it is no longer clear how to construct them efficiently, which we leave as a future challenge.

Acknowledgements We thank the anonymous reviewers for their valuable feedback. This work has been supported by the EPSRC through grants EP/X021513/1 and EP/X017796/1.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Inf. Comput. **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6, https://doi.org/10.1016/0890-5401(87)90052-6
2. Angluin, D., Boker, U., Fisman, D.: Families of DFAs as Acceptors of ω-Regular Languages. Logical Methods in Computer Science **14**(1) (2018)
3. Angluin, D., Fisman, D.: Learning regular omega languages. Theor. Comput. Sci. **650**, 57–72 (2016). https://doi.org/10.1016/j.tcs.2016.07.031, https://doi.org/10.1016/j.tcs.2016.07.031
4. Angluin, D., Fisman, D.: Regular ω-languages with an informative right congruence. Inf. Comput. **278**, 104598 (2021). https://doi.org/10.1016/j.ic.2020.104598, https://doi.org/10.1016/j.ic.2020.104598

[6] https://github.com/iscas-tis/roll-library

5. Bohn, L., Löding, C.: Passive learning of deterministic Büchi automata by combinations of DFAs. In: Bojanczyk, M., Merelli, E., Woodruff, D.P. (eds.) 49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France. LIPIcs, vol. 229, pp. 114:1–114:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ICALP.2022.114, https://doi.org/10.4230/LIPIcs.ICALP.2022.114
6. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960. pp. 1–12. Stanford University Press (1962)
7. Calbrix, H., Nivat, M., Podelski, A.: Ultimately periodic words of rational w-languages. In: Brookes, S.D., Main, M.G., Melton, A., Mislove, M.W., Schmidt, D.A. (eds.) Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993, Proceedings. Lecture Notes in Computer Science, vol. 802, pp. 554–566. Springer (1993). https://doi.org/10.1007/3-540-58027-1_27, https://doi.org/10.1007/3-540-58027-1_27
8. Colcombet, T., Zdanowski, K.: A tight lower bound for determinization of transition labeled büchi automata. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II. Lecture Notes in Computer Science, vol. 5556, pp. 151–162. Springer (2009). https://doi.org/10.1007/978-3-642-02930-1_13, https://doi.org/10.1007/978-3-642-02930-1_13
9. Esparza, J., Kretínský, J., Raskin, J., Sickert, S.: From LTL and limit-deterministic büchi automata to deterministic parity automata. In: Legay, A., Margaria, T. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10205, pp. 426–442 (2017). https://doi.org/10.1007/978-3-662-54577-5_25, https://doi.org/10.1007/978-3-662-54577-5_25
10. Farzan, A., Chen, Y., Clarke, E.M., Tsay, Y., Wang, B.: Extending automated compositional verification to the full class of omega-regular languages. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4963, pp. 2–17. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_2, https://doi.org/10.1007/978-3-540-78800-3_2
11. Krishnan, S.C., Puri, A., Brayton, R.K.: Deterministic w automata vis-a-vis deterministic buchi automata. In: Du, D., Zhang, X. (eds.) Algorithms and Computation, 5th International Symposium, ISAAC '94, Beijing, P. R. China, August 25-27, 1994, Proceedings. Lecture Notes in Computer Science, vol. 834, pp. 378–386. Springer (1994). https://doi.org/10.1007/3-540-58325-4_202, https://doi.org/10.1007/3-540-58325-4_202
12. Kurshan, R.P.: Complementing deterministic büchi automata in polynomial time. J. Comput. Syst. Sci. **35**(1), 59–71 (1987). https://doi.org/10.1016/0022-0000(87)90036-5, https://doi.org/10.1016/0022-0000(87)90036-5

13. Li, Y., Chen, Y., Zhang, L., Liu, D.: A novel learning algorithm for büchi automata based on family of dfas and classification trees. Inf. Comput. **281**, 104678 (2021). https://doi.org/10.1016/j.ic.2020.104678, https://doi.org/10.1016/j.ic.2020.104678
14. Li, Y., Sun, X., Turrini, A., Chen, Y., Xu, J.: ROLL 1.0: \omega -regular language learning library. In: Vojnar, T., Zhang, L. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11427, pp. 365–371. Springer (2019). https://doi.org/10.1007/978-3-030-17462-0_23, https://doi.org/10.1007/978-3-030-17462-0_23
15. Li, Y., Turrini, A., Feng, W., Vardi, M.Y., Zhang, L.: Divide-and-conquer determinization of büchi automata based on SCC decomposition. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13372, pp. 152–173. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_8, https://doi.org/10.1007/978-3-031-13188-2_8
16. Maler, O., Pnueli, A.: On the learnability of infinitary regular sets. Inf. Comput. **118**(2), 316–326 (1995). https://doi.org/10.1006/inco.1995.1070, https://doi.org/10.1006/inco.1995.1070
17. Maler, O., Staiger, L.: On syntactic congruences for omega-languages. Theor. Comput. Sci. **183**(1), 93–112 (1997). https://doi.org/10.1016/S0304-3975(96)00312-X, https://doi.org/10.1016/S0304-3975(96)00312-X
18. Michel, M.: Complementation is more difficult with automata on infinite words. CNET, Paris **15** (1988)
19. Myhill, J.: Finite automata and the representation of events. In: Technical Report WADD TR-57-624. p. 112–137 (1957)
20. Nerode, A.: Linear automaton transformations. In: American Mathematical Society. p. 541–544 (1958)
21. Pfleeger, C.P.: State reduction in incompletely specified finite-state machines. IEEE Trans. Computers **22**(12), 1099–1102 (1973). https://doi.org/10.1109/T-C.1973.223655, https://doi.org/10.1109/T-C.1973.223655
22. Safra, S.: On the complexity of omega-automata. In: 29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988. pp. 319–327. IEEE Computer Society (1988). https://doi.org/10.1109/SFCS.1988.21948, https://doi.org/10.1109/SFCS.1988.21948
23. Schewe, S.: Tighter bounds for the determinisation of büchi automata. In: de Alfaro, L. (ed.) Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5504, pp. 167–181. Springer (2009). https://doi.org/10.1007/978-3-642-00596-1_13, https://doi.org/10.1007/978-3-642-00596-1_13
24. Schewe, S.: Beyond hyper-minimisation—minimising dbas and dpas is np-complete. In: Lodaya, K., Mahajan, M. (eds.) IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India. LIPIcs, vol. 8, pp. 400–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik

(2010). https://doi.org/10.4230/LIPIcs.FSTTCS.2010.400, https://doi.org/10.4230/LIPIcs.FSTTCS.2010.400

25. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification (preliminary report). In: Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986. pp. 332–344. IEEE Computer Society (1986)
26. Wilke, T., Schewe, S.: ω-automata. In: Pin, J. (ed.) Handbook of Automata Theory, pp. 189–234. European Mathematical Society Publishing House, Zürich, Switzerland (2021). https://doi.org/10.4171/Automata-1/6, https://doi.org/10.4171/Automata-1/6

A Proof of Lemma 3

Lemma 3. *Let $\Sigma_n = \{0, 1, \cdots, n\}$. There exists an ω-regular language L_n over Σ_n such that its limit FDFA has $\Theta(n)$ states, while both its syntactic and recurrent FDFAs have $\Theta(n^2)$ states.*

Proof. The language L_n is given as its DBA $\mathcal{B} = (Q, \Sigma_n, q_0, \delta, \Gamma)$ depicted in Figure 2. First, we show that the index of $\backsim_{L_n}$ is $n+2$. In fact, the leading DFA induced by $\backsim_{L_n}$ is the exactly the TS of $\mathcal{B}$. For every two words $u_1, u_2 \in \Sigma^*$, if $u_1 \not\backsim_{L_n} u_2$, then there exists a word $w \in \Sigma^\omega$ such that $u_1 \cdot w \in L_n \Longleftrightarrow u_2 \cdot w \in L_n$ does not hold. That is, $u_1^{-1} \cdot L_n \neq u_2^{-1} \cdot L_n$ where $u^{-1} \cdot L_n = \{w \in \Sigma^\omega : u \cdot w \in L_n\}$ for a word $u \in \Sigma^*$. Let $L_q = \mathcal{L}(\mathcal{B}^q)$. For every pair of different states $q_i, q_j \in Q$ with $i \neq j$, obviously $L_{q_i} \neq L_{q_j}$ since L_{q_i} contains an infinite word i^ω, while L_{q_j} does not contain such a word. So, if $\mathcal{B}(u_1) \neq \mathcal{B}(u_2)$, then $u_1^{-1} \cdot L_n \neq u_2^{-1} \cdot L_n$. Hence, $|\backsim_{L_n}| \geq n + 2$. It is trivial to see that $|\backsim_{L_n}| \leq n + 2$ since the index of $\backsim_{L_n}$ is always not greater than the number of states in a deterministic ω-automaton accepting L_n. Therefore, $|\backsim_{L_n}| = n + 2$.

Now we fix a word u and consider the index of $\approx^u_L$. Let $x \in \Sigma^*$. Obviously, if $q_\perp = \mathcal{B}(u)$, then for all $v \in \Sigma^*$, we have $u \cdot x \cdot v \backsim_{L_n} u$ but $u \cdot (x \cdot v)^\omega \notin L_n$. Hence, $|\approx^u_L| = 1$. Now let $q_i = \mathcal{B}(u)$ with $0 \leq i \leq n$. For all $v \in \Sigma^*$, if $u \cdot x \cdot v \backsim_{L_n} u$ holds, it must be the case that $u \cdot (x \cdot v)^\omega \in L_n$ except that $x \cdot v = \epsilon$. Hence, $|\approx^u_L| = 2$. It follows that the limit FDFA of L_n has exactly $2 \times (n + 1) + 1 + n + 2 \in \Theta(n)$ states.

Now we consider the index of $\approx^u_R$ for a fixed $u \in \Sigma^*$. Similarly, when $q_\perp = \mathcal{B}(u)$, $|\approx^u_R| = 1$ since for all $v \in \Sigma^*$, we have $u \cdot x \cdot v \backsim_{L_n} u \wedge u \cdot (x \cdot v)^\omega \notin L_n$ hold. Now we consider that $q_k = \mathcal{B}(u)$ with $0 \leq k \leq n$. Let $x_1, x_2 \in \Sigma^*$. First, assume that $\mathcal{B}(u \cdot x_1) \neq \mathcal{B}(u \cdot x_2)$. W.l.o.g., let $q_j = \mathcal{B}(u \cdot x_2)$ with $0 \leq j \leq n$ and let $q_i = \mathcal{B}(u \cdot x_1)$ with either $i < j$ or $q_i = q_\perp$. We can easily construct a finite word v such that $q_k = \mathcal{B}(u) = \mathcal{B}(u \cdot x_2 \cdot v)$, i.e., $u \cdot x_2 \cdot v \backsim_{L_n} u$, and $u \cdot (x_2 \cdot v)^\omega \in L_n$. For example, we can let $v = (j + 1) \cdots n \cdot 0 \cdots k$ if $j < k \leq n$. Hence, $u \cdot x_2 \cdot v \backsim_{L_n} u \wedge u \cdot (x_2 \cdot v)^\omega \in L_n$ holds. On the contrary, it is easy to see that $q_\perp = \mathcal{B}(u \cdot x_1 \cdot v) = \delta(q_i, j + 1)$ since either $j + 1 > i + 1$ or $q_i = q_\perp$. In other words, we have $u \cdot x_1 \cdot v \not\backsim_{L_n} u \wedge u \cdot (x_1 \cdot v)^\omega \notin L_n$ holds. By definition of $\approx^u_R$, $x_1 \not\approx^u_R x_2$. Hence, $|\approx^u_R| \geq n + 2$. Next, we assume that $\mathcal{B}(u \cdot x_1) = \mathcal{B}(u \cdot x_2)$. For a word $v \in \Sigma^*$, it is easy to see that $u \cdot x_1 \cdot v \backsim_{L_n} u \Longleftrightarrow u \cdot x_2 \cdot v \backsim_{L_n} u$. Moreover, since $u \cdot x_1 \cdot v \backsim_{L_n} u$ implies $u \cdot (x_1 \cdot v)^\omega \in L_n$, we thus have that $u \cdot x_1 \cdot v \backsim_{L_n} u \wedge u \cdot (x_1 \cdot v)^\omega \in L_n \Longleftrightarrow u \cdot x_2 \cdot v \backsim_{L_n} u \wedge u \cdot (x_2 \cdot v)^\omega \in L_n$. In other words, $x_1 \approx^u_R x_2$, which implies that $|\approx^u_R| \leq n + 2$. Hence $|\approx^u_R| = n + 2$ when $\mathcal{B}(u) \neq q_\perp$. It follows that the recurrent FDFA of L_n has exactly $(n + 2) \times (n + 1) + 1 + (n + 2) \in \Theta(n^2)$ states.

For the syntactic FDFA, since $\approx^u_S$ refines $\approx^u_R$ [3], then $|\approx^u_S| \geq |\approx^u_R|$ for all $u \in \Sigma^*$. The upper bound is proved similarly as for recurrent FDFAs. Therefore, the syntactic FDFA of L_n also has $\Theta(n^2)$ states.

This completes the proof of the lemma. □

B Translations from FDFAs to NBAs

It is possible to transform a canonical FDFA $\mathcal{F}$ of L to an equivalent NBA $\mathcal{A}$ [2, 7, 13].

In the following, we only berifly describe how we construct a NBA from an FDFA. Angluin and Fisman proved in [2] that every saturated FDFA $\mathcal{F}$ can be polynomially translated to an equivalent NBA $\mathcal{A}[\mathcal{F}]$. In fact, the requirement for $\mathcal{F}$ being saturated is somewhat strong; we only need $\mathcal{F}$ to be almost saturated.

The translation given in [2, 7, 13] works as follows. Let $\mathcal{F} = (\mathcal{M}, \{\mathcal{N}^q\})$ be an almost saturated FDFA, where $\mathcal{M} = (\Sigma, Q, \iota, \delta)$, and for each state $q \in Q$, there is a progress DFA $\mathcal{N}^q = (\Sigma, Q_q, \iota_q, \delta_q, F_q)$. Recall that $(A)^s_f$ denotes the DFA A where s is the initial state and f is the sole final state. By Definition 3, we have that $\mathrm{UP}(\mathcal{F}) = \{\alpha \in \Sigma^\omega : \alpha \textit{ is accepted by } \mathcal{F}\}$, where α is accepted if there is a decomposition (u, v) of α, such that $\mathcal{M}(u) = \mathcal{M}(uv)$, and $\mathcal{N}^q(v) \in F_q$ where $q = \mathcal{M}(u)$. This implies that a word $\alpha \in \mathrm{UP}(\mathcal{F})$ can be decomposed into two parts u and v, such that u is accepted by the DFA $\mathcal{M}^\iota_q$ and v by the DFA $(\mathcal{N}^q)^{\iota_q}_f$ where $f = \mathcal{N}^q(v)$. Hence, $\mathrm{UP}(\mathcal{F}) = \bigcup_{q \in Q, f \in F_q} \mathcal{L}_*(M^\iota_q) \cdot N_{(q,f)}$, where $N_{(q,f)} = \{v^\omega \in \Sigma^\omega : v \in \Sigma^+, q = \mathcal{M}^q_q(v), v \in \mathcal{L}_*((\mathcal{N}^q)^{\iota_q}_q)\}$ is the set of all infinite repetitions of the finite words v accepted by $(\mathcal{N}^q)^{\iota_q}_f$.

It is hard to construct a NBA to accept exactly $N_{(q,f)}$. However, it suffices to under approximate $N_{(q,f)}$ with the DFA $P_{(q,f)} = \mathcal{M}^q_q \times (\mathcal{N}^q)^{\iota_q}_q \times (\mathcal{N}^q)^f_f$, where $\times$ stands for the intersection product between DFAs. On one hand, the DFA $\mathcal{M}^q_q \times (\mathcal{N}^q)^{\iota_q}_q$ makes sure that for a word $v \in \mathcal{L}_*(\mathcal{M}^q_q \times (\mathcal{N}^q)^{\iota_q}_q)$ and $u \in \mathcal{L}_*(\mathcal{M}^\iota_q)$, it follows that $q = \mathcal{M}(u) = \mathcal{M}(uv)$. On the other hand, $(\mathcal{N}^q)^f_f$ ensures that $v, v^k \in \mathcal{L}_*((\mathcal{N}^q)^{\iota_q}_f)$ for all $k \geq 1$. One can construct a NBA $\mathcal{A}[\mathcal{F}] = \bigcup_{q \in Q, f \in F_q} \mathcal{L}_*(\mathcal{M}^\iota_q) \cdot P^\omega_{(q,f)}$ to under approximate $\mathrm{UP}(\mathcal{F})$ [13].

It is worth noting that we can construct easily a DBA that accepts $P^\omega_{(q,f)}$ from the DFA $P_{(q,f)}$ by redirecting all incoming transitions of final states to the initial state and mark them as Γ-transitions. This way, we obtain a LDBA $\mathcal{S}[\mathcal{F}]$ that recognizes $\mathrm{UP}(\mathcal{F})$, which allows easier determinization algorithm [9, 15]. This construction of LDBAs is much easier than the one proposed in [13] where the acceptance condition is defined on states, rather than transitions.

Since the four types of canonical FDFAs are all saturated, Corollary 5 immediately follows.

Corollary 5. *Let L be an ω-regular language. Then its periodic, syntactic, recurrent and limit FDFAs are almost saturated.*

Let n is the number of states in the leading DFA $\mathcal{M}$ and k is the largest number of states of progress DFAs of $\mathcal{F}$. For each pair $q \in Q, f \in F_q$, the constructed NBA/DBA accepting $P_{(q,f)}$ has nk^2 states, and there are at most nk such pairs; So, all four types of canonical FDFAs can be polynomial translated to equivalent NBA/LDBAs with $\mathcal{O}(n^2k^3)$ states.

For the variant limit FDFA $\mathcal{F}_B$, there is at most one final state in each progress DFA. So, the equivalent NBA for $\mathcal{F}_B$ has $\mathcal{O}(n^2k^2)$ states.

C Proof of Lemma 4

Lemma 4. *Let L be a DBA-recognizable language and $\mathcal{F}_L{=}(\mathcal{M}, \{\mathcal{N}^u_L\}_{[u]_\backsim \in \Sigma^*/_\backsim})$ be the limit FDFA of L. Then, for each progress DFA $\mathcal{N}^u_L$ with $\mathcal{L}_*(\mathcal{N}^u_L) \neq \emptyset$, there must exist a final state $\tilde{x} \in F_u$ such that $[\tilde{x}]_{\approx^u_L} = \{x \in \Sigma^+ : \forall v \in \Sigma^*.\ u \cdot (x \cdot v) \backsim u \implies u \cdot (x \cdot v)^\omega \in L\}$.*

Proof. The proof is inspired and adapted from the proof of [5, Lemma 10].

We let $\mathcal{D} = (\mathcal{T}, \Gamma)$ be a DBA of L, where $\mathcal{T} = (Q, \Sigma, q_0, \delta)$ is the TS of $\mathcal{D}$ and Γ is the set of accepting transitions. We assume that $\mathcal{D}$ is complete in the sense that for every state $q \in Q$ and $\sigma \in \Sigma$, we have that $\delta(q, \sigma) \in Q$.

For two different states $q_1, q_2 \in Q$, we define an equivalence relation $\backsim_\mathcal{D}$ where $q_1 \backsim_\mathcal{D} q_2$ if and only if $\mathcal{L}(\mathcal{D}^{q_1}) = \mathcal{L}(\mathcal{D}^{q_2})$ where $\mathcal{D}^q$ is the DBA obtained from $\mathcal{D}$ by setting the initial state to $q \in Q$. Let $U_q = \{u \in \Sigma^* : \delta(q_0, u) = q\}$. Let $U_{[q]_{\backsim_\mathcal{D}}} = \cup_{p \in [q]_{\backsim_\mathcal{D}}} U_p$ where $[q]_{\backsim_\mathcal{D}}$ is the equivalence class of $\backsim_\mathcal{D}$ that q belongs to. Clearly, $U_{[q]_{\backsim_\mathcal{D}}}$ is an equivalence class $[u]_\backsim$ of $\backsim$ defined with respect to L where $u \in U_{[q]_{\backsim_\mathcal{D}}}$.

Now consider the periodic finite words for each state $q \in Q$. Let $V_q = \{x \in \Sigma^+ : \forall v \in \Sigma^*.\ \text{if } q \xrightarrow{x \cdot v} q.\ (x \cdot v)^\omega \in \mathcal{L}(\mathcal{D}^q)\}$. That is, a word x belongs to V_q iff for every $v \in \Sigma^*$, if $\mathcal{D}$ takes a round trip from q back to itself over $x \cdot v$, the run must go through a Γ-transition. We first prove that V_q is regular. We can construct the DFA D_q of V_q from the TS $\mathcal{T}$ by first removing all Γ-transitions in $\mathcal{T}$, resulting a TS $\mathcal{T}'$, and then collect all the transitions (p, σ, q) in a set β such that p and q are in the different SCCs of the reduced TS $\mathcal{T}'$. We then define $D_q = (Q \cup \{\top\}, \Sigma, q, \delta_D, F = \{\top\})$ where (1) for a state $p \in Q$, $\sigma \in \Sigma$ and $q = \delta(p, \sigma)$, $\delta_D(p, \sigma) = q$ if $(p, \sigma, q) \notin \Gamma \cup \beta$ and otherwise $\delta_D(p, \sigma) = \top$; and (2) $\delta_D(\top, \sigma) = \top$ for all $\sigma \in \Sigma$.

Next we prove that $\mathcal{L}_*(D_q) = V_q$. First, let $x \in \mathcal{L}_*(D_q)$ and we want to prove that $x \in V_q$. Obviously, the last transition of $\mathcal{D}$ over x from q will be either a Γ-transition or a transition jumping between two SCCs in the reduced $\mathcal{T}'$. If it is a Γ-transition, obviously, we have that for all $v \in \Sigma^*$, if $q \xrightarrow{x \cdot v} q$, then it must visit a Γ-transition. Hence, $(xv)^\omega \in \mathcal{L}(\mathcal{D}^q)$. If it is a transition jumping between different SCCs, it would be the case that either $\mathcal{D}$ does not go back to q over xv or it must be visiting a Γ-transition, since in the reduced TS $\mathcal{T}'$, they can not reach each other. Therefore, $x \in V_q$. Now let $x \in V_q$ and we want to prove that $x \in \mathcal{L}_*(D_q)$. Let $p = \delta(q, x)$ in $\mathcal{D}$. If p and q lie in two different SCCs of $\mathcal{D}$, then it is impossible to find a $v \in \Sigma^*$ such that $p \xrightarrow{v} q$, otherwise, p and q will belong to the same SCC of $\mathcal{D}$. In this case, there will be a transition between different SCCs along the way from q to p over xv, which of courses also separates these two SCCs in the reduced TS $\mathcal{T}'$. Thus, there will be a prefix of x accepted by D_q, so x is also accepted by D_q as $\top$ is a sink final state. Now assume that p and q are in the same SCC of $\mathcal{D}$. At state p, for each $v \in \Sigma^*$ such that $q \xrightarrow{x} p \xrightarrow{v} q$, we have that $(x \cdot v)^\omega \in \mathcal{L}(\mathcal{D}^q)$. There must be some Γ-transition visited along the way from q back to itself. It follows that in the reduced TS $\mathcal{T}'$, it is impossible to reach p from q. In other words, q and p are not in the same SCC of $\mathcal{T}'$. So, the run from q to p over x must visit some transition jumping between two different

SCCs. Again, this means that there will be a prefix of x accepted by D_q. So x will also be accepted by D_q. Therefore, V_q is a regular language.

Now, for an equivalence class $[q]_{\backsim_\mathcal{D}}$, we define $V_{[q]_{\backsim_\mathcal{D}}} = \bigcap_{p\in[q]_{\backsim_\mathcal{D}}} V_p$. So, $V_{[q]_{\backsim_\mathcal{D}}}$ is also a regular language. Let u be a word in $U_{[q]_{\backsim_\mathcal{D}}}$.

Let $V_u = \{x \in \Sigma^+ : \forall v \in \Sigma^*.\ u \cdot (x \cdot v) \backsim u \implies u \cdot (x \cdot v)^\omega \in L\}$. Next, we prove that $V_u \equiv V_{[q]_{\backsim_\mathcal{D}}}$. Let $p = \delta(q_0, u)$.

Let $x \in V_{[q]_{\backsim_\mathcal{D}}}$ and we want to prove that $x \in V_u$. That is, we need to prove that for all $v \in \Sigma^*$, we have that $u \cdot (x \cdot v) \backsim u \implies u \cdot (x \cdot v)^\omega \in L$. First, if $u \cdot (x \cdot v) \not\backsim u$, then $x \in V_u$ holds trivially. Otherwise we have that $u \cdot x \cdot v \backsim u$, which implies that $\delta(q_0, u \cdot (x \cdot v)^k) \backsim_\mathcal{D} \delta(q_0, u)$ for all $k \geq 0$. Thus, we will have a run $\rho = q_0 \xrightarrow{u} q_1 \xrightarrow{x\cdot v} \cdots$ of $\mathcal{D}$ over $u \cdot (xv)^\omega$ where $q_i \in [q]_{\backsim_\mathcal{D}}$ for all $i > 0$. There must be some state q occurs for an inifinite set of indices $I = \{i \in \mathbb{N} : q = q_i\}$. For each $q_i \in [q]_{\backsim_\mathcal{D}}$, we have that $x \in V_{q_i}$. First, $x \in V_p$ for all states $p \in [q]_{\backsim_\mathcal{D}}$, so for every two pairs of integers $i, j \in I$ with $i < j$, there must be a Γ-transition along the way from q_i to q_j. It follows that $u \cdot (x \cdot v)^\omega \in \mathcal{L}(\mathcal{D}^q)$ holds. Hence, $x \in V_u$ holds as well, since $u \cdot x \cdot v \backsim u \implies u \cdot (x \cdot v)^\omega \in L$ holds for all $v \in \Sigma^*$.

Now assume that $x \notin V_{[q]_{\backsim_\mathcal{D}}}$ and we want to prove that $x \notin V_u$ holds. Assume by contradiction that $x \in V_u$. Since x does not belong to $V_{[q]_{\backsim_\mathcal{D}}}$, then there exists a state $r \in [q]_{\backsim_\mathcal{D}}$ such that $x \notin V_r$. That is, there exists a word $v \in \Sigma^*$ such that $r \xrightarrow{x\cdot v} r$ and $(x \cdot v)^\omega \notin \mathcal{L}(\mathcal{D}^r)$. Since $p \backsim_\mathcal{D} r$, i.e., $\mathcal{L}(\mathcal{D}^p) = \mathcal{L}(\mathcal{D}^r)$, $(x \cdot v)^\omega \notin \mathcal{L}(\mathcal{D}^p)$ as well. It then follows that $u \cdot (x \cdot v) \backsim u$ and $u \cdot (x \cdot v)^\omega \notin L$, which contradicts that $x \in V_u$. Therefore, $x \notin V_u$.

Hence, $V_u = V_{[q]_{\backsim_\mathcal{D}}}$. Now we show that V_u is an equivalence class of $\approx^u_L$ as follows. On one hand, for every two different words $x_1, x_2 \in V_u$, we have that $x_1 \approx^u_L x_2$, which is obvious by the definition of V_u. On the other hand, it is easy to see that $x' \not\approx^u_L x$ for all $x' \notin V_u$ and $x \in V_u$ because there will exists some $v \in \Sigma^*$ such that $u \cdot x' \cdot v \backsim u$ but $u \cdot (x' \cdot v)^\omega \notin L$. Hence, V_u is indeed an equivalence class of $\approx^u_L$. Obviously, $V_u \subseteq \mathcal{L}_*(\mathcal{N}^u)$, as we can let $v = \epsilon$, so for every word $x \in V_u$, we have that $u \cdot x \backsim u \implies u \cdot x^\omega \in L$. Let $\tilde{x} = \mathcal{N}^u(x)$ for a word $x \in V_u$. It follows that $\tilde{x}$ is a final state of $\mathcal{N}^u$ and we have $[\tilde{x}]_{\approx^u_L} = V_u$. Thus, we complete the proof of the lemma.

□

D Proof of Theorem 6

Theorem 6. *Let L be an ω-regular language and $\mathcal{F}_L{=}(\mathcal{M}[\backsim], \{\mathcal{N}[\approx_u]\}_{[u]_\backsim \in \Sigma^*/_\backsim})$ be the limit FDFA of L s.t. $\approx_u \in \mathsf{RC}(\approx^u_N)$ with finite index for all u. Then (1) $\mathcal{F}_L$ has a finite number of states, (2) UP($\mathcal{F}_L$) = UP(L), and (3) $\mathcal{F}_L$ is saturated.*

Proof. The first claim follows from the restriction to finite indices in the definition (we have seen that they exist, and that we can, e.g., choose limit RC).

To show $\mathrm{UP}(\mathcal{F}_L) \subseteq \mathrm{UP}(L)$, assume that $w \in \mathrm{UP}(\mathcal{F}_L)$. By Definition 3, a UP-word w is accepted by $\mathcal{F}_L$ if there exists a decomposition (u, v) of w such that $\mathcal{M}(u) = \mathcal{M}(u \cdot v)$ (equivalently, $u \cdot v \backsim u$) and $v \in \mathcal{L}_*(\mathcal{N}^{\tilde{u}})$ where $\tilde{u} = \mathcal{M}(u)$.

Here $\tilde{u}$ is the representative word for the equivalence class $[u]_{\backsim}$. Similarly, let $\tilde{v} = \mathcal{N}^{\tilde{u}}(v)$. By Definition 12, we have $\tilde{u} \cdot \tilde{v} \backsim \tilde{u} \implies \tilde{u} \cdot \tilde{v}^{\omega} \in L$ holds as $\tilde{v}$ is a final state of $\mathcal{N}^{\tilde{u}}$. Since $v \approx_{\tilde{u}} \tilde{v}$ (i.e., $\mathcal{N}^{\tilde{u}}(v) = \mathcal{N}^{\tilde{u}}(\tilde{v})$), $\tilde{u} \cdot v \backsim \tilde{u} \implies \tilde{u} \cdot v^{\omega} \in L$ holds as well. It follows that $u \cdot v \backsim u \implies u \cdot v^{\omega} \in L$ since $u \backsim \tilde{u}$ and $u \cdot v \backsim \tilde{u} \cdot v$ (equivalently, $\mathcal{M}(u \cdot v) = \mathcal{M}(\tilde{u} \cdot v)$). Together with the assumption that $\mathcal{M}(u \cdot v) = \mathcal{M}(u)$ (i.e, $u \backsim u \cdot v$), we then have that $u \cdot v^{\omega} \in L$ holds. So, $\mathrm{UP}(\mathcal{F}_L) \subseteq \mathrm{UP}(L)$ also holds.

To show that $\mathrm{UP}(L) \subseteq \mathrm{UP}(\mathcal{F}_L)$ holds, let $w \in \mathrm{UP}(L)$. For a UP-word $w \in L$, we can find a normalized decomposition (u, v) of w such that $w = u \cdot v^{\omega}$ and $u \cdot v \backsim u$ (i.e., $\mathcal{M}(u) = \mathcal{M}(u \cdot v)$), since the index of $\backsim$ is finite (cf. [3] for more details). Let $\tilde{u} = \mathcal{M}(u)$ and $\tilde{v} = \mathcal{N}^{\tilde{u}}(v)$. Our goal is to prove that $\tilde{v}$ is a final state of $\mathcal{N}^{\tilde{u}}$. Since $u \backsim \tilde{u}$ and $u \cdot v^{\omega} \in L$, then $\tilde{u} \cdot v^{\omega} \in L$ holds. Moreover, $\tilde{u} \cdot v \backsim \tilde{u}$ holds as well because $\tilde{u} = \mathcal{M}(\tilde{u}) = \mathcal{M}(u) = \mathcal{M}(\tilde{u} \cdot v) = \mathcal{M}(u \cdot v)$. (Recall that $\mathcal{M}$ is deterministic.) We now have that $v \in C_u$, so that $C_{\tilde{u}} \cap \Sigma^* /_{\approx^u_N}$ is good (as $u \cdot v^{\omega} \in L$). We also have that $\tilde{v} \approx^u_N v$, so that $[\tilde{v}]_{\approx^u_N}$ is accepting. Hence, $\tilde{v}$ is a final state, and (u, v) therefore accepted by $\mathcal{F}_L$, i.e., $w \in \mathrm{UP}(\mathcal{F}_L)$. It follows that $\mathrm{UP}(L) \subseteq \mathrm{UP}(\mathcal{F}_L)$.

Now we prove that $\mathcal{F}_L$ is saturated. Let w be a UP-word. Let (u, v) and (x, y) be two normalized decompositions of w with respect to $\mathcal{M}$ (or, equivalently, to $\backsim$). We have seen that (u, v) is accepted by $\mathcal{F}_L$ iff $u \cdot v^{\omega} = x \cdot y^{\omega} \in \mathrm{UP}(L)$, which is the case iff (x, y) is accepted by $\mathcal{F}_L$ with the same argument. □

E Active learning of limit FDFAs

First, there are two roles, namely the learner and an oracle in the active learning framework [1]. The task of the learner is to learn an automaton representation of an unknown language L from the oracle. The learner can ask two types of queries about L, which will be answered by the oracle. A membership query is about whether a word w is in L; an equivalence query is to ask whether a given automaton recognizes the language L. If the oracle returns positive answer to equivalence query, then the learner has completed the task and output the correct automaton; otherwise, the learner will receive a counterexample which will then be used to refine current hypothesis.

Angluin and Fisman proposed a learning framework in [3] to learn the classical three types of FDFAs. We show that our limit FDFA can easily fit into this learning framework. The learner L^{ω} is described in the following framework. We refer to [3] for details about the learning framework. We mainly use the notations and description from [3] in the following. As usual, the framework makes use of the notion of *observation tables*. An observation table is a tuple $\mathcal{T} = (S, \tilde{S}, E, T)$ where S is a prefix-closed set of finite words, E is a set of experiments trying to distinguish the strings in S, and $T : S \times E \to D$ stores the element (membership query results) in entry $T(s, e)$ an element in some domain D, where $s \in S$ and $e \in E$. For our limit FDFA, D is purely a Boolean values $\{\top, \bot\}$. We usually determine when two strings $s_1, s_2 \in S$ should be considered not equivalent depending on the RC we are using. The component $\tilde{S} \subseteq S$ is the subset considered as representatives of the equivalence classes, i.e., the state names of the constructed DFA. A table is said to be *closed* if S is prefix closed and for every $s \in \tilde{S}$ and $\sigma \in \Sigma$, we have $s\sigma \in S$. The procedure *CloseTable* uses two sub-procedures ENT and DFR to make a given observation closed. Here ENT is used to fill in the entries of the table by means of asking membership queries. The procedure DFR is used to determine which row (words) of the table should be distinguished. A learning procedure usually begins with create an initial observation table by asking membership queries, close the table with ENT and DFR procedures, and then construct an hypothesis automaton for asking equivalence query. The learner should be able to use the counterexample to the equivalence query to find new experiments for discovering new equivalence classes.

We now give the subprocedures for learning our limit FDFAs.

Algorithm 1: The learner L^ω in [3]

```
Initialize leading table 𝒯 = (S, S̃, E, T) with
  S = S̃ = {ϵ}, E = {(ϵ, σ) : σ ∈ Σ};
CloseTable(𝒯, ENT_1, DFR_1) and let ℳ = Aut_1(𝒯);
forall u ∈ S̃ do
    Initialize 𝒯_u = (S_u, S̃_u, E_u, T_u), with S_u = S̃_u = E_u = {ϵ};
    CloseTable(𝒯_u, ENT_2^u, DFR_2^u) and let 𝒜_u = Aut_2(𝒯_u);
while true do
    Let (a, u, v) be the oracle's response for equivalence query ℋ = (ℳ, {𝒜_u});
    if a = "yes" then
        break;
    Let (x, y) be the normalized decomposition of (u, v) w.r.t ℳ;
    Let x̃ = ℳ(x);
    if MQ(x, y) ≠ MQ(x̃, y) then
        E = E ∪ FindDistinguishingExperiment(x, y);
        CloseTable(𝒯, ENT_1, DFR_1) and let ℳ = Aut_1(𝒯);
    else
        E_x̃ = E_x̃ ∪ FindDistinguishingExperiment(x̃, y);
        CloseTable(𝒯_x̃, ENT_2^x̃, DFR_2^x̃) and let 𝒜_x̃ = Aut_2(𝒯_x̃);
```

We let $\mathtt{MQ}(x, y)$ be the result of the membership query ω-word $x \cdot y^\omega$ to the oracle. The procedures $\mathtt{ENT}_1$ and $\mathtt{DFR}_1$ and $\mathtt{Aut}_1$ are the same for all four types of FDFAs. More precisely, for $u, x, y \in \Sigma^*$, $\mathtt{ENT}_1(u, (x, y)) = \mathtt{MQ}(u \cdot x, y)$; for two finite row words $u_1, u_2 \in S$, $\mathtt{DFR}_1(u_1, u_2) = \top$ iff there exists $(x, y) \in E$ such that $T(u_1, (x, y)) \neq T(u_2, (x, y))$. That is, we can use $x \cdot y^\omega$ to distinguish the finite words u_1 and u_2 according to $\backsim$. The procedure $\mathtt{Aut}_1$ is simply to construct the leading DFA without final states from $\mathcal{T}$, by Definition 11. When learning our limit FDFAs, for $u, x, v \in \Sigma^*$, we define $\mathtt{ENT}_2^u(x, v) = \top$ if $\mathcal{M}(ux \cdot v) \neq \mathcal{M}(u)$ or $\mathtt{MQ}(u, x \cdot v) = \top$ holds, corresponding to whether

$ux \cdot v \backsim u \implies u \cdot (xv)^\omega \in L$ holds in Definition 9; for two finite row words, $x_1, x_2 \in S_u$, $\mathtt{DFR}_2^u(x_1, x_2)$ returns true if there exists $v \in E$ such that $T_u(x_1, v) \neq T_u(x_2, v)$. The procedure $\mathtt{Aut}_u(\mathcal{T}_u)$ not only constructs the TS but also set a state x as accepting if $T_u(x, \epsilon) = \top$. Note that here $T_u(x, v)$ stores the result of whether $(\mathcal{M}(u \cdot xv) = \mathcal{M}(u)) \implies \mathtt{MQ}(u, xv)$.

To be consistent with the notations in [3], we also denote by $\rho[i..k]$ the subsequence of ρ starting at the i-th element and ending at the k-th element (inclusively) when $i \leq k$, and the empty sequence ϵ when $i > k$. However, the first element will be $\rho[1]$ instead of $\rho[0]$ in the main content.

Now we provide more details in learning our limit FDFAs and also prove that the learner L^ω will make progress in every iteration. We assume that now we have received the counterexample (u, v) in the algorithm to current hypothesis and we prove that our limit FDFA learner is able to make use of (u, v) to refine current FDFA.

Let (x, y) be the normalized decomposition of the counterexample $u \cdot v^\omega$ with respect to $\mathcal{M}$ and let $\tilde{x} = \mathcal{M}(x)$. If $\mathtt{MQ}(x, y) \neq \mathtt{MQ}(\tilde{x}, y)$, then we know that $x \not\backsim \tilde{x}$. So, we can find an experiment as follows: let $n = |x|$ and for $1 \leq i \leq n$, let $s_i = \mathcal{M}(x[1 \cdots i])$ be state/word that $\mathcal{M}$ arrives after reading the first i letters of x. Recall that s_i is also the representative word of $\mathcal{M}(x[1 \cdots i])$. In particular, $s_0 = \mathcal{M}(\epsilon) = \epsilon$ and $s_n = \mathcal{M}(x) = \tilde{x}$. Thus, we can construct the sequence, $\mathtt{MQ}(s_0 \cdot x[1 \cdots n], y), \mathtt{MQ}(s_1 \cdot x[2 \cdots n], y), \mathtt{MQ}(s_2 \cdot x[3 \cdots n], y), \cdots, \mathtt{MQ}(s_n \cdot x[n+1 \cdots n], y)$. Obviously, this sequence has different results for the first and last elements since $\mathtt{MQ}(s_0 \cdot x[1 \cdots n], y) \neq \mathtt{MQ}(s_n, y)$, where $s_n = \tilde{x}$.

Therefore, there must exist the smallest $j \in [1 \cdots n]$ such that $\mathtt{MQ}(s_{j-1} \cdot x[j \cdots n], y) \neq \mathtt{MQ}(s_j \cdot x[j+1 \cdots n], y)$, It follows that we can use the experiment $e = (u[j+1 \cdots n], v)$ to distinguish $s_{j-1} \cdot x[j]$ and s_j.

Otherwise if $\mathtt{MQ}(x, y) = \mathtt{MQ}(\tilde{x}, y)$, we need to similarly refine current $\mathcal{A}_{\tilde{x}}$. Similarly, we let $n = |y|$ and $s_i = \mathcal{A}_{\tilde{x}}(y[1 \cdots i])$. We also consider a sequence $(m_0, c_0), \cdots, (m_n, c_n)$ where $m_i = \top$ iff $\tilde{x} = \mathcal{M}(\tilde{x} \cdot s_i \cdot y[i+1 \cdots n])$ and $c_i = \top$ iff $\tilde{x} \cdot (s_i \cdot y[i+1 \cdots n])^\omega \in L$. First, we know that $m_0 = \top$ and $m_n = \top$ since (x, y) is a normalized decomposition of $u \cdot v^\omega$, i.e., $\tilde{x} = \mathcal{M}(x) = \mathcal{M}(x \cdot y) = \mathcal{M}(\tilde{x} \cdot y)$. Since (x, y) is a counterexample to current hypothesis $\mathcal{H}$, we know that either the normalized decomposition (x, y) is not accepted by $\mathcal{H}$ and $xy^\omega \in L$ or (x, y) is accepted by $\mathcal{H}$ and $xy^\omega \notin L$. Therefore, one out of (m_0, c_0) and (m_n, c_n) must be $(\top, \top)$ and the other is not. That is, either $m_0 \implies c_0$ or $m_n \implies c_0$ holds. There must be the smallest $j \in [1 \cdots n]$ such that $m_{j-1} \implies c_{j-1}$ and $m_j \implies c_j$ differs. W.l.o.g., we let $m_{j-1} \implies c_{j-1}$ hold. In this case, we can set the experiment $e = y[j+1 \cdots n]$ to distinguish $s_{j-1} \cdot y[j]$ and s_j since we have $\tilde{x} = \mathcal{M}(\tilde{x} \cdot s_{j-1} \cdot y[j \cdots n]) \implies \tilde{x} \cdot (s_{j-1} \cdot y[j \cdot n])^\omega \in L$ but $\tilde{x} = \mathcal{M}(\tilde{x} \cdot s_j \cdot y[j+1 \cdots n]) \implies \tilde{x} \cdot (s_j \cdot y[j+1 \cdots n])^\omega \in L$ does not hold.

We can see that every time we received a counterexample from the oracle, either the leading DFA $\mathcal{M}$ or the progress DFA $\mathcal{A}_{\tilde{x}}$ will add at least state. Since the limit FDFA $\mathcal{F}_L$ has finite number of states, $\mathcal{H}$ will eventually be $\mathcal{F}_L$ in the worst case.

Corollary 6. *The limit FDFAs can be learned with membership and equivalence queries in time in polynomial in the size of canonical limit FDFAs.*