A Novel Family of Finite Automata for Recognizing and Learning ω-Regular Languages

Yong Li, Sven Schewe and Qiyi Tang

University of Liverpool

Angluin-Style learning framework

Learning languages via membership and equivalence queries

Applications to verification:

- Assumptions for compositional verification [Cobleigh et al 2003]
- Automata model for neural networks [Xu et al. 2021 ; Muškardin et al. 2022]

Foundation of Angluin-Style learning

Myhill-Nerode theorem for a language R over Σ:

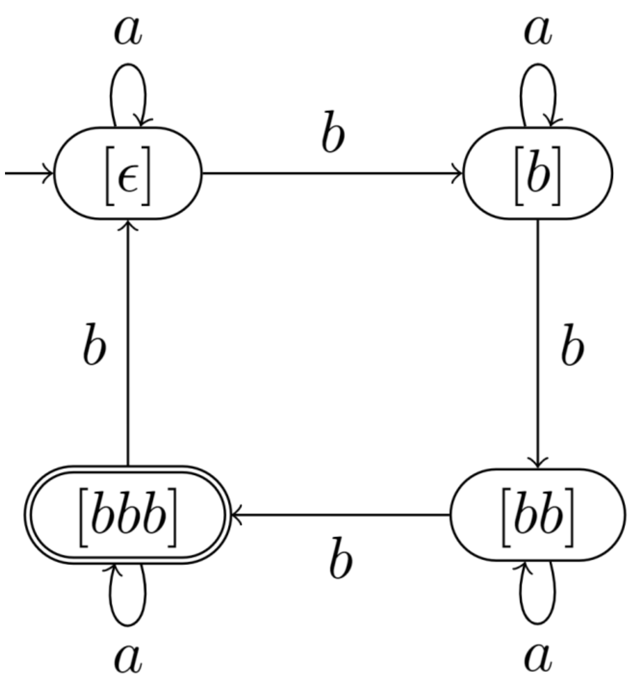
$u_1 \sim u_2$ iff for all $v \in \Sigma^*. u_1 v \in R \iff u_2 v \in R$

- R is regular iff the number of equivalence classes (equivalently, states) of $\sim$ is finite
- Defines the minimal DFA of R

Minimal DFA example

Myhill-Nerode theorem defines the minimal DFA:

$u_1 \sim u_2$ iff for all $v \in \Sigma^*. u_1 v \in R \iff u_2 v \in R$



$R = \{u \in \Sigma^* |$ the number of $b's$ in u is $4n + 3$ for $n \in \mathbb{N}\}$

ϵ and b can be distinguished with $v = bb$:

- $\epsilon \cdot bb \notin R$
- $b \cdot bb \in R$

What about ω-regular languages

- Simple extension does not work for an ω-language L
  - $u_1 \sim u_2$ iff for all $w \in \Sigma^\omega . u_1 \cdot w \in L \iff u_2 \cdot w \in L$
- No canonical forms of deterministic automata with Büchi, **Muller**, **Rabin, Parity** and **Streett** conditions
- Not easy: minimization is **NP**-complete [Schewe 2010]

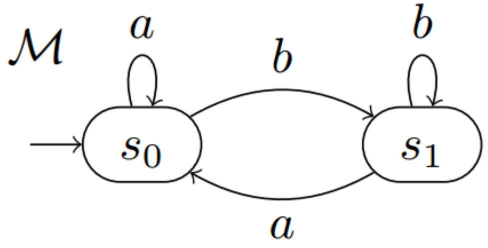
What about ω-regular languages

- ω-regular expression $L = U_1 \cdot V_1^\omega + \cdots + U_n \cdot V_n^\omega$ where we have regular languages $U_i \subseteq \Sigma^*, V_i \subseteq \Sigma^+$ for $1 \leq i \leq n$

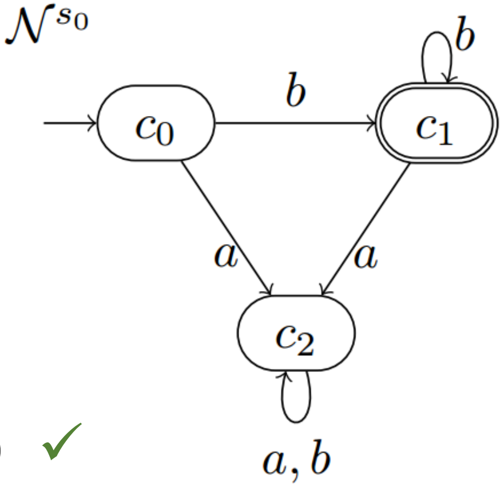
- How about just the ultimately periodic (UP)-words (u, v) s.t. $u \in U_i$ and $v \in V_i$

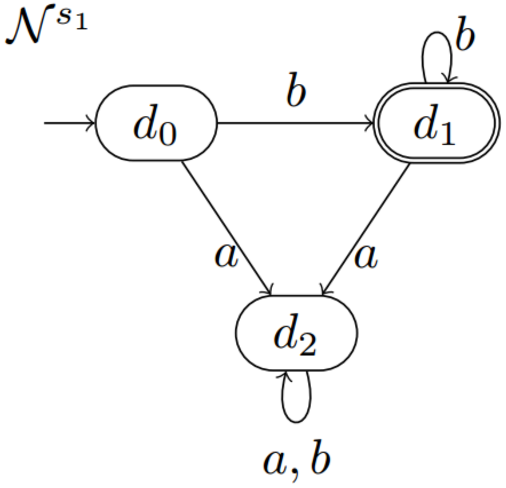
Family of DFAs

- **Family of DFAs** (FDFAs) [Angluin, Boker & Fisman' 16]
  - **Leading** DFA for prefixes u
  - **Progress** DFAs for periodic words v
  - Accept (u, v) if $M(u) = M(uv)$ and $v \in L(N^{M(u)})$



- Normalized decompositions (u, v):
  - $M(u) = M(uv)$





- Example FDFA for $L = \Sigma^* \cdot b^\omega$
  - Run over $(ab, b) = a \cdot b^\omega$ is $(s_0 s_0 s_1, d_0 d_1)$ ✓
  - Run over $(a, b) = a \cdot b^\omega$ is $(s_0 s_0, d_0 d_1)$ ✗

Why FDFAs

- Myhill-Nerode theorem for canonical FDFAs
- L is ω-regular iff the number of states in its FDFA is finite
- Application to learning ω-regular languages

Canonical FDFAs

- **Periodic**, **Syntactic**, and **Recurrent** FDFAs
- Recurrent FDFAs are more succinct
- Complexity of learning is polynomial in size

Our contributions

Novel canonical form called **Limit FDFA**:

- Dual to Recurrent FDFAs, more succinct than others
- L is ω-regular iff its number of states is finite
- Easy to decide DBA-recognizable languages

Canonical FDFAs

Canonical FDFA $F = (M, \{N^u\})$ for an ω-language L

Leading DFA M for processing the finite prefix:

- $u_1 \sim u_2$ iff for all $w \in \Sigma^\omega . u_1 \cdot w \in L \iff u_2 \cdot w \in L$

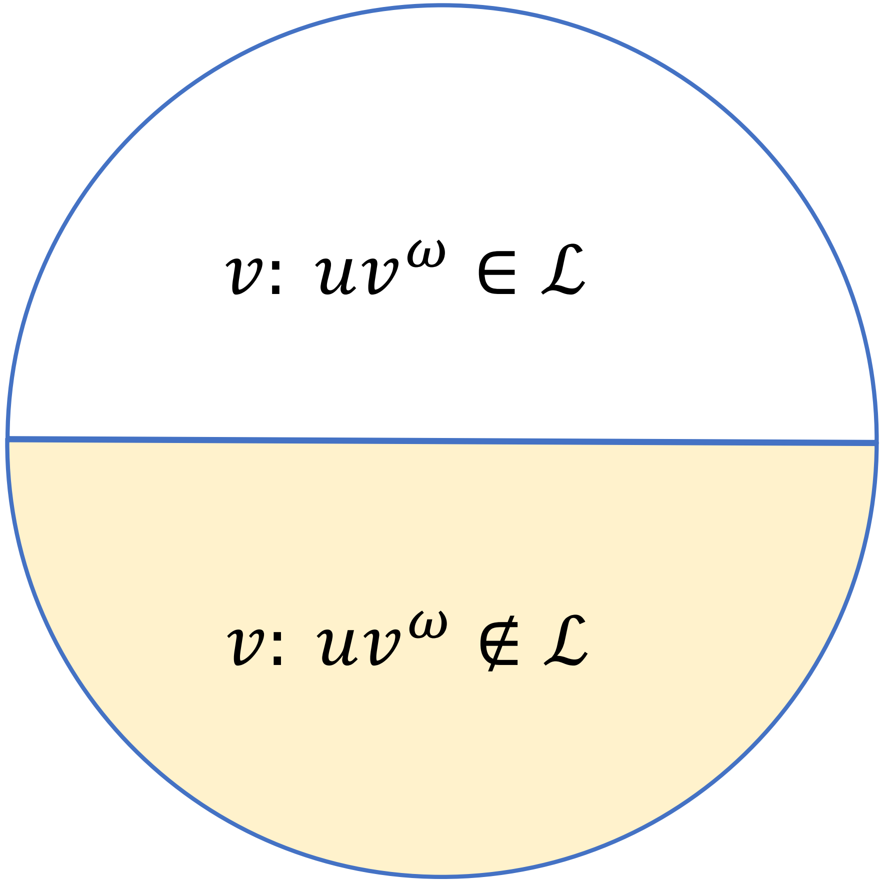
Normalized decomposition: $M(u) = M(uv)$ iff $u \sim uv$

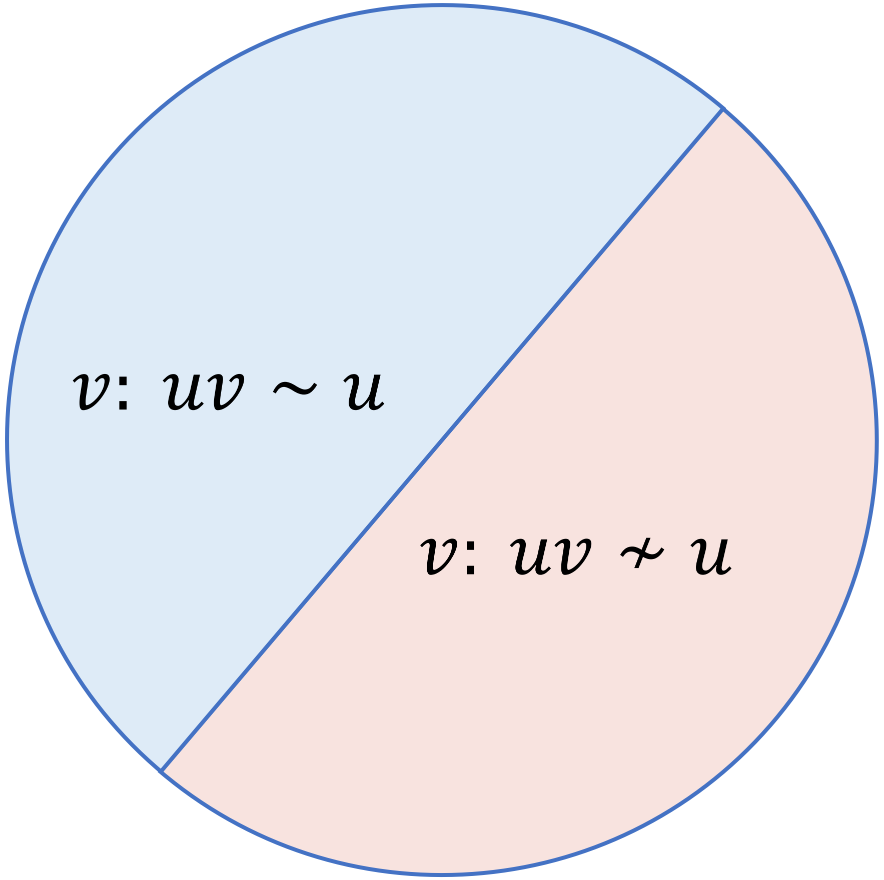
Progress DFA N^u for accepting periodic words $v \in \Sigma^*$:

- Similar to $v_1 \approx v_2$ iff for all $y \in \Sigma^* . v_1 \cdot y \in V \iff v_2 \cdot y \in V$
- Vary on the progress language $V = \mathbf{L}(N^u)$

Ways to partition periodic words

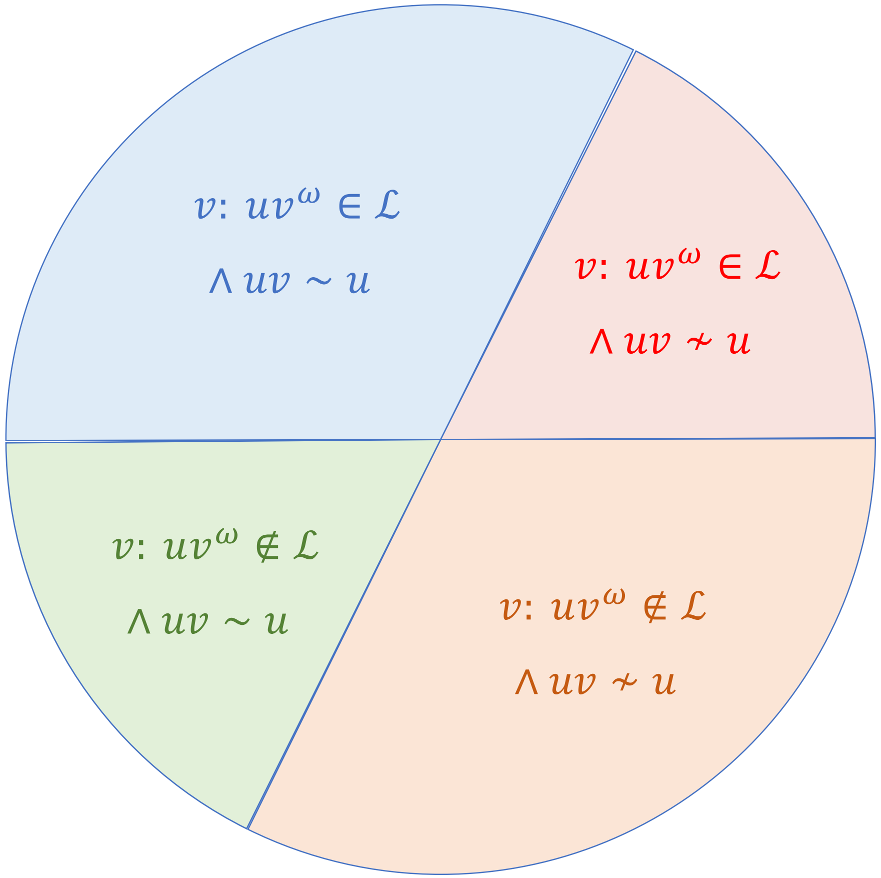
Fix a $u \in \Sigma^*$, two ways to partition periodic words in Σ^*





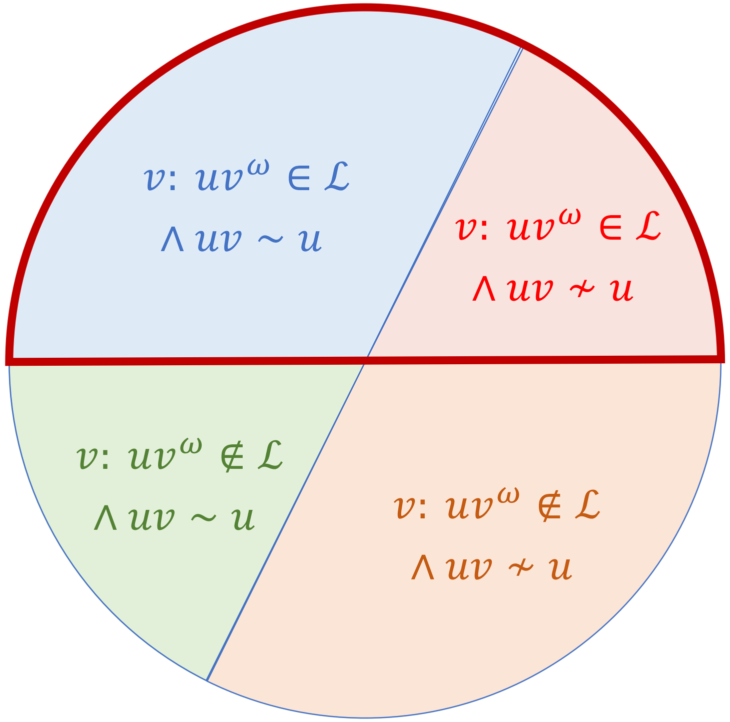
Ways to partition periodic words

Fix a $u \in \Sigma^*$, four blocks for periodic words in Σ^*



Progress languages for different FDFAs

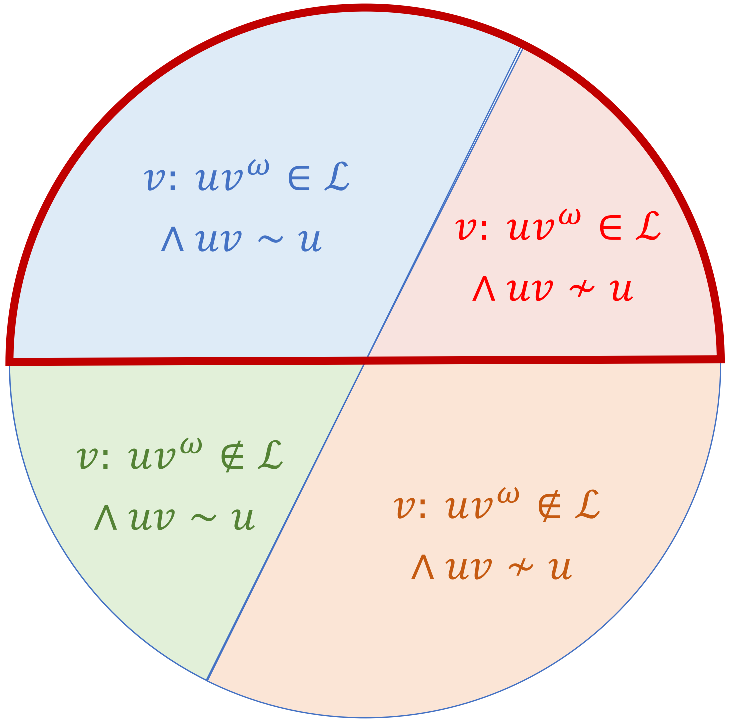
Fix a $u \in \Sigma^*$, the progress language V in different FDFAs



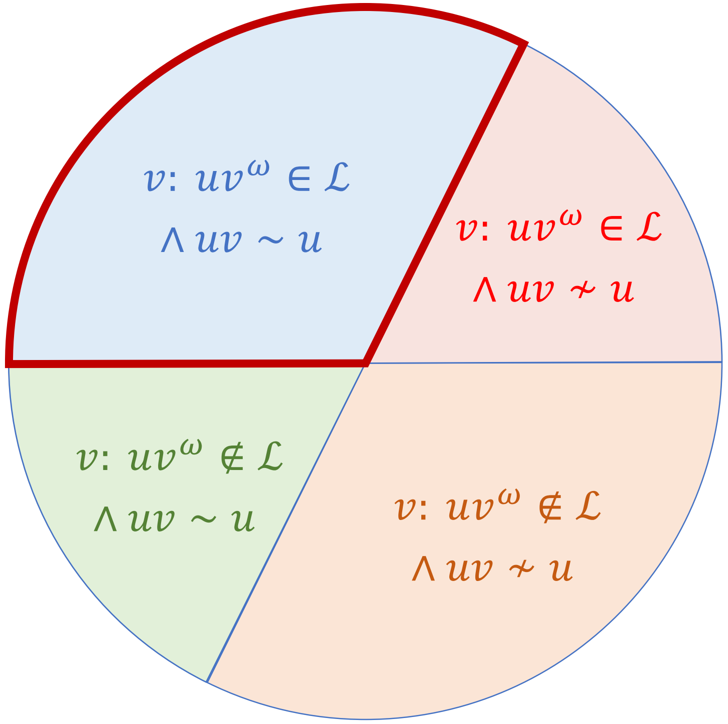
Periodic FDFA

Progress languages for different FDFAs

Fix a $u \in \Sigma^*$, the progress language V in different FDFAs



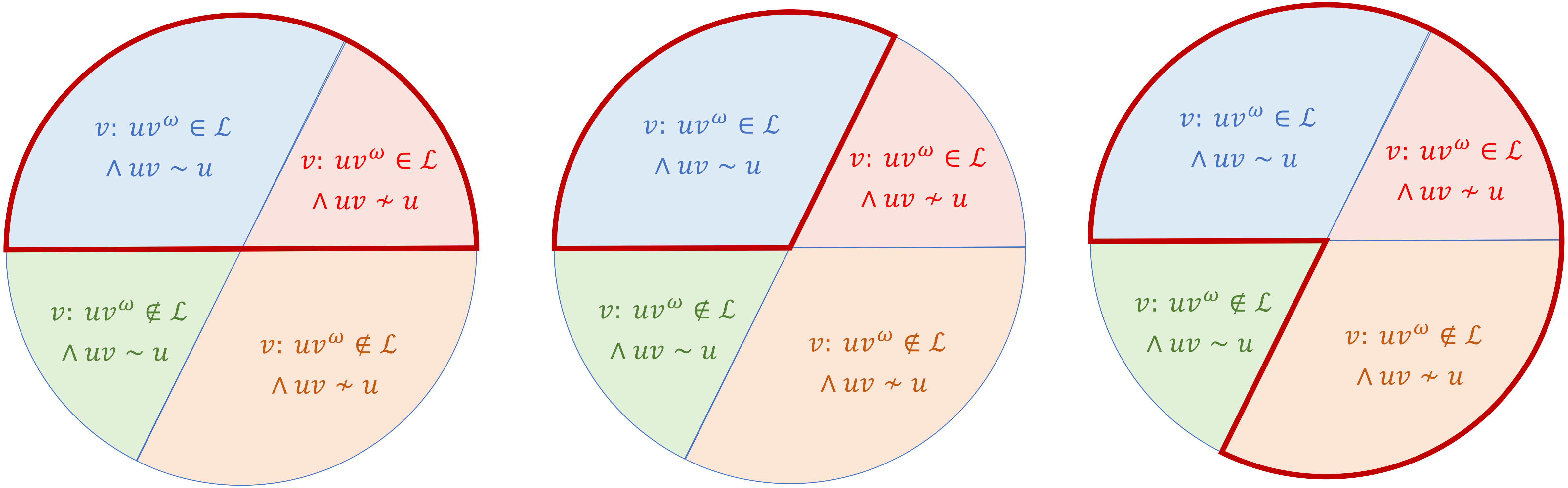
Periodic FDFA



Syntactic/Recurrent FDFA

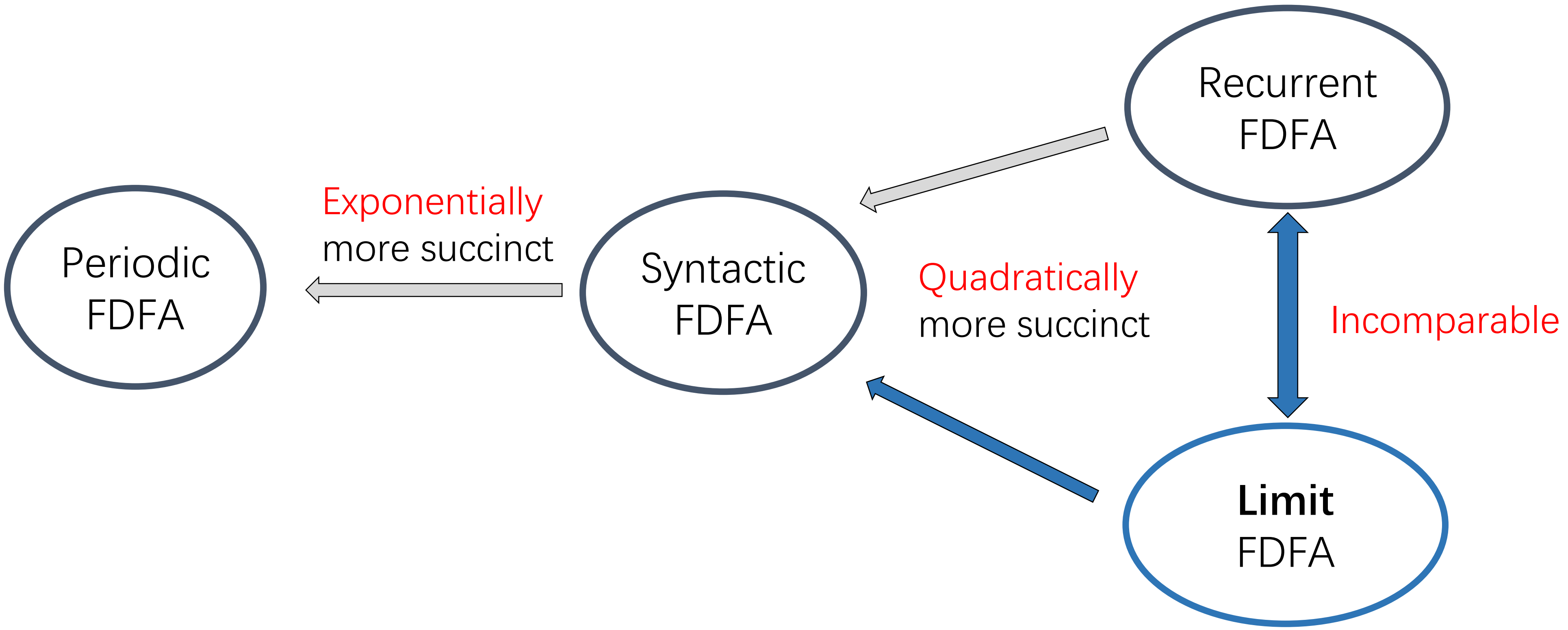
Progress languages for different FDFAs

Fix a $u \in \Sigma^*$, the progress language V in different FDFAs



Periodic FDFA　　Syntactic/Recurrent FDFA　　**Limit** FDFA

Limit FDFAs vs other FDFAs in size



Myhill-Nerode theorem with Limit FDFAs

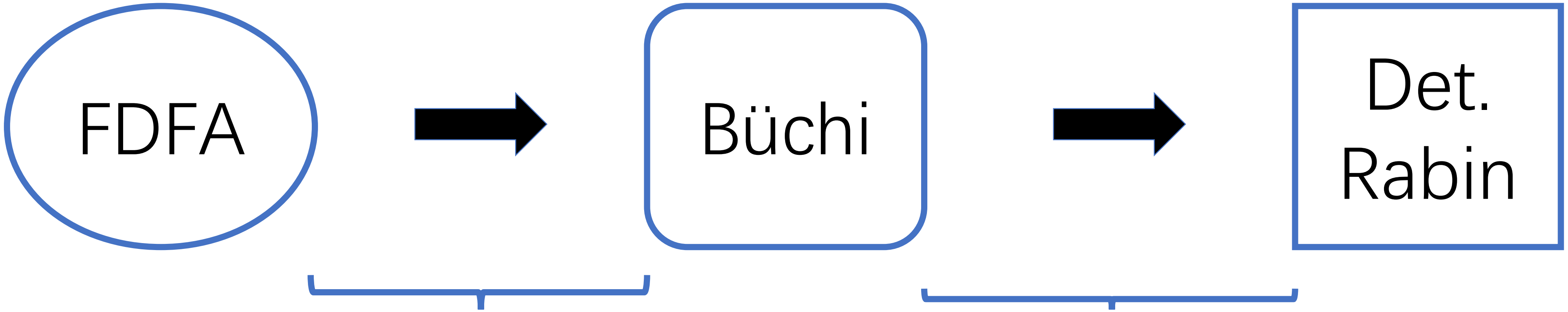
Our "Myhill-Nerode" Theorem: For an ω-language L, L is ω-regular iff the number of states in its limit FDFA is finite

Proof idea:

- Limit FDFAs are more succinct than Syntactic FDFAs but only quadratically more succinct
- The Myhill-Nerode theorem with **Syntactic** FDFAs [Maler & Staiger' 97]

Theorem 1: For an ω-regular language L, the limit FDFA of L recognizes L correctly

Deciding DBA-recognizable languages

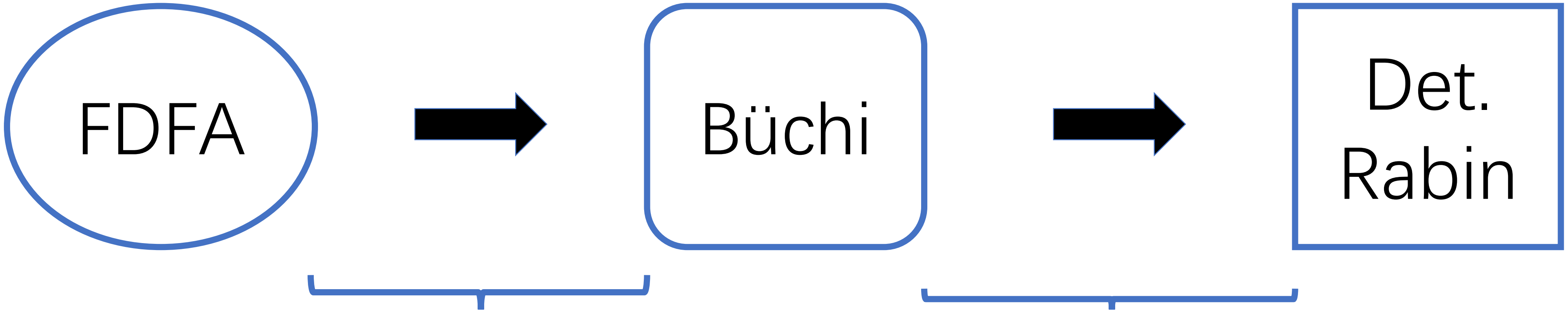


$O(n^3)$ blow-up
[Calbrix, Nivat & Podelski 1994]

$\Omega(1.64n)^n$ in determinization
[Colcombet & Zdanowski 2009]

Decide DBA-recognizability
[Krishnan, Puri & Brayton 1994]

Deciding DBA-recognizable languages



$O(n^3)$ blow-up
[Calbrix, Nivat & Podelski 1994]

$\Omega(1.64n)^n$ in determinization
[Colcombet & Zdanowski 2009]

Exponential blow-up

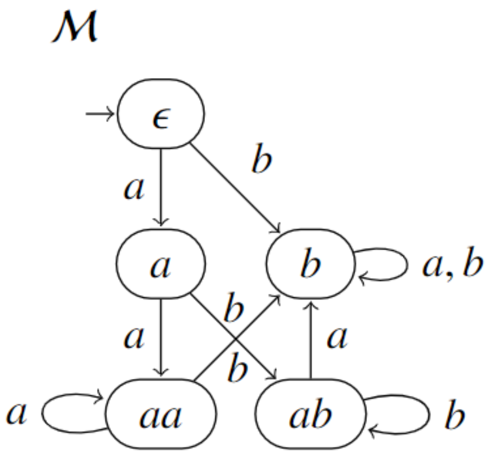
Decide DBA-recognizability
[Krishnan, Puri & Brayton 1994]

Observations on DBA languages

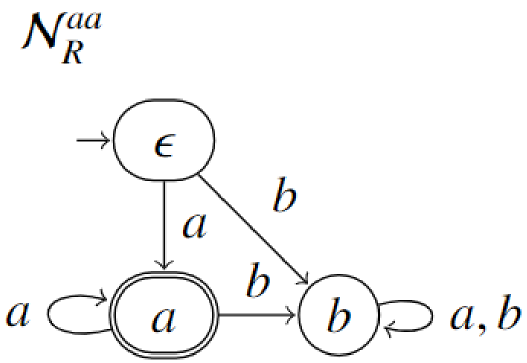
Lemma 1: For a DBA language L, every progress DFA in its **limit** FDFA either has a sink final state or no final states at all

Example language $L = a^\omega + ab^\omega$

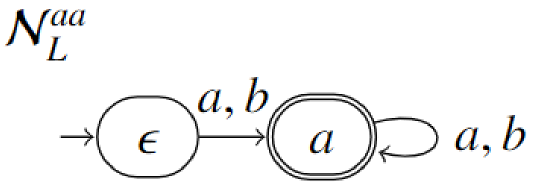
Leading DFA



Recurrent progress DFA for aa



Limit progress DFA for aa



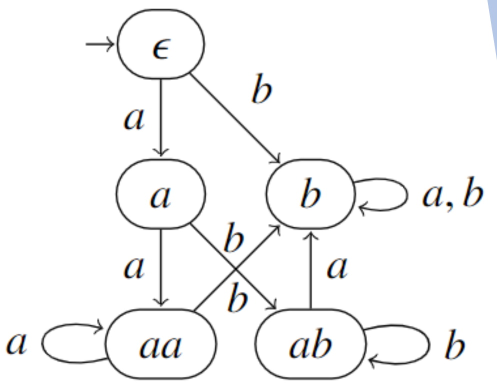
Observations on DBA languages

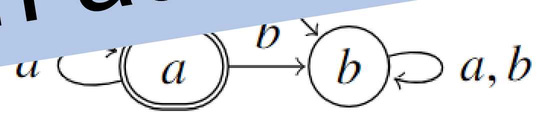
Lemma 1: For a DBA language L, every progress DFA in its **limit** FDFA either has a sink final state or no final states at all

Example language $L = \ a^\omega + ab^\omega$

Leading DFA

$\mathcal{M}$

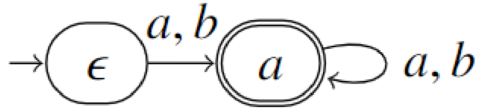




Accept all words that do not loop from aa

Limit progress DFA for aa

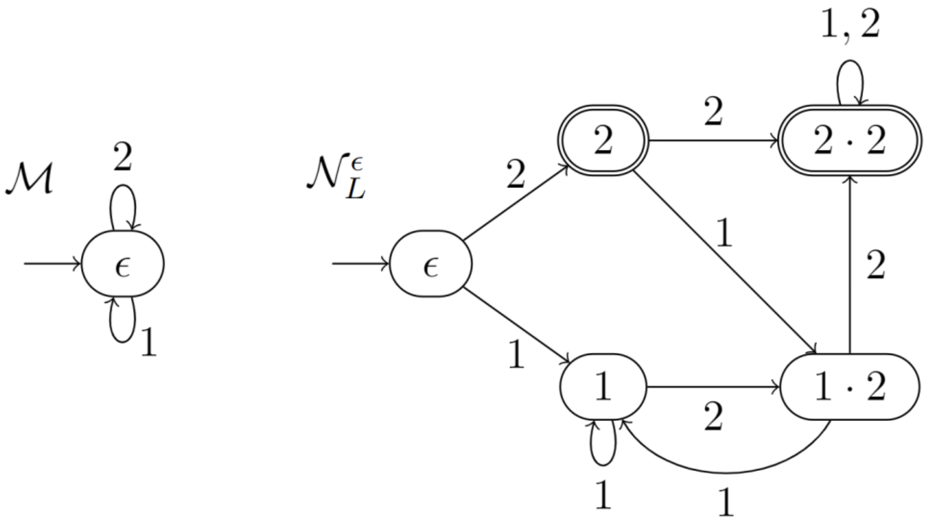
$\mathcal{N}_L^{aa}$



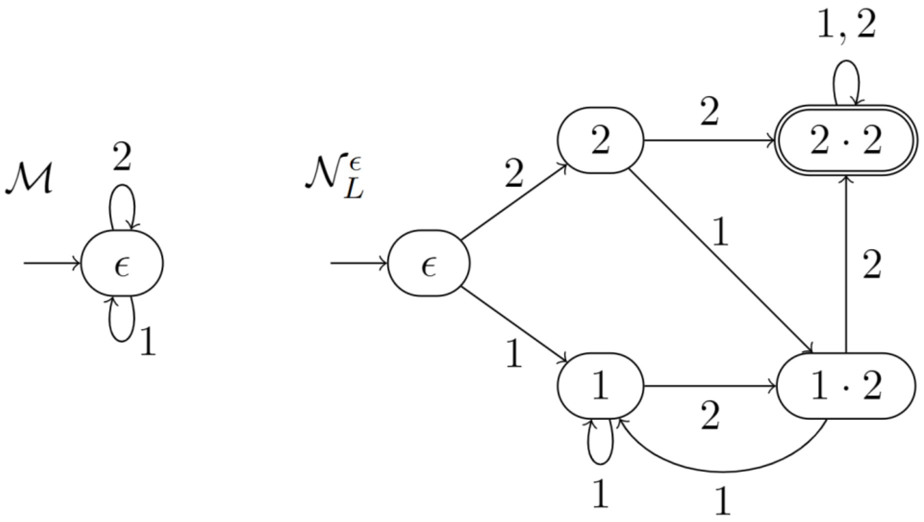
Limit FDFAs for DBA languages

Theorem 2: Sink final states suffice iff it is DBA language

DBA-recognizable: $L = (\{1,2\}^{*} \cdot (2 \cdot 2))^{\omega}$



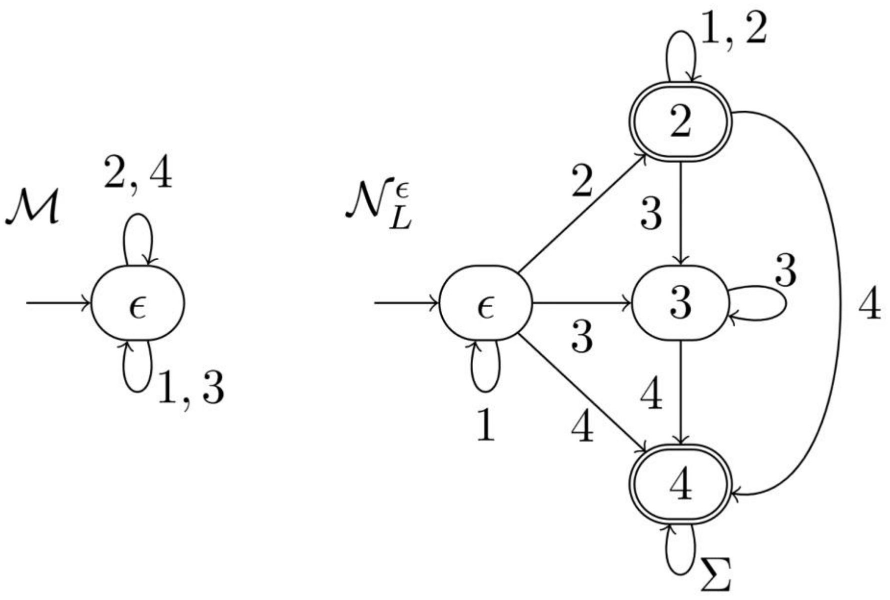
Unmark
non-sink
final states



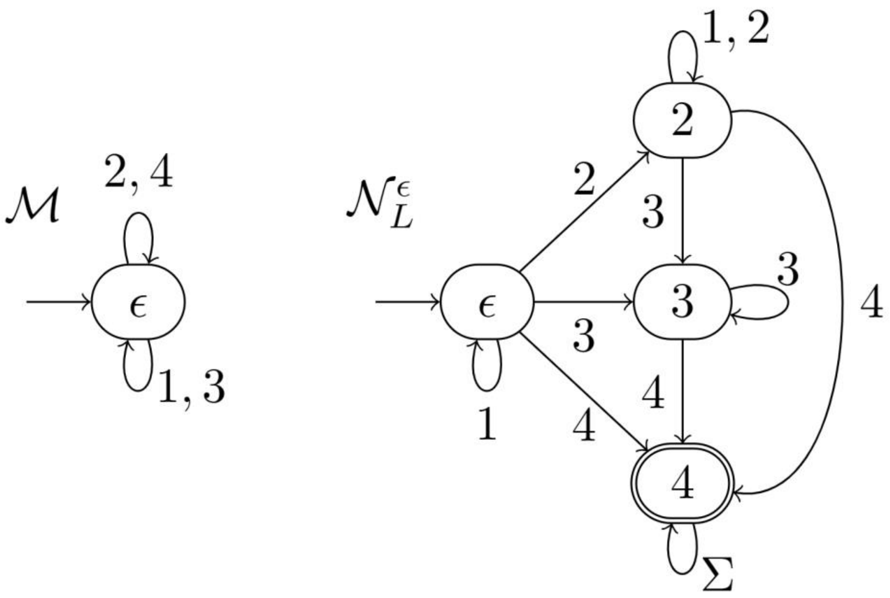
Limit FDFAs for DBA languages

Theorem 2: Sink final states suffice iff it is DBA language

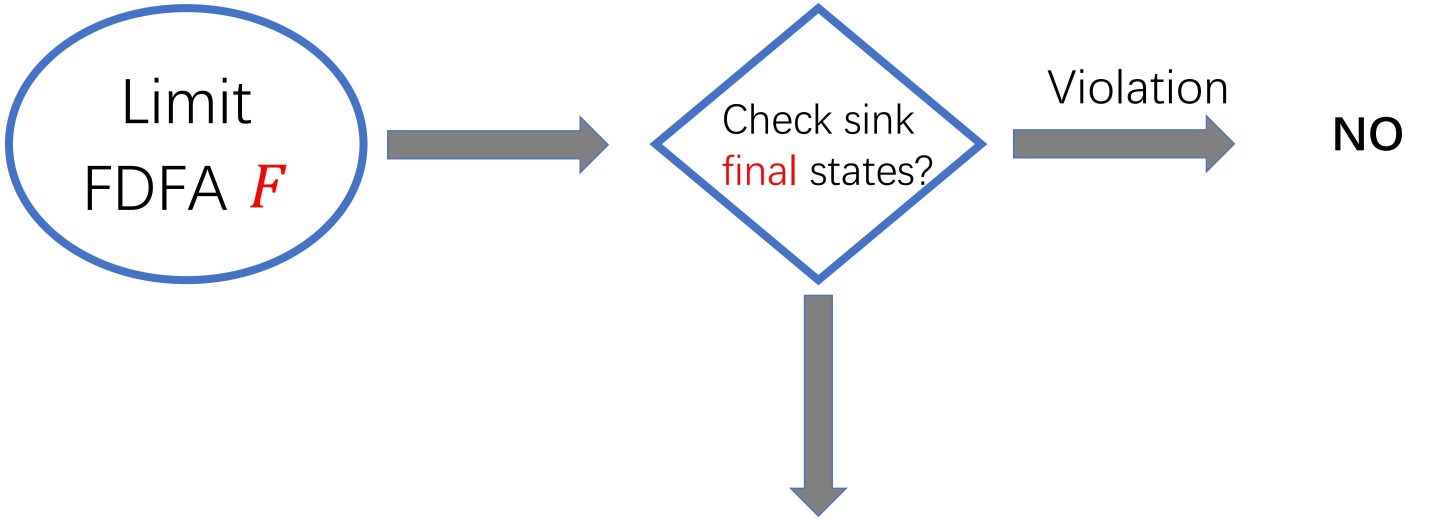
Not DBA-recognizable: $L = \{w \in \Sigma^{\omega} \mid \max \inf(w) \text{ is even}\}$



Unmark
non-sink
final states



Deciding DBA-recognizable languages



Sink final states retain languages?

1. Unmark non-sink final states and obtain F'
2. Check containment between NBA(F) and DBA(F')
3. If no words are missing, return **YES**, otherwise **NO**

Summary

- Novel canonical form: **Limit FDFAs**
- Myhill-Nerode theorem using Limit FDFAs
- Polynomial decision procedure for DBA-languages
- Requirements to define minimal progress DFAs

- Future work:
  - Empirical evaluation for learning ω-regular languages
  - Learning DBAs as representation

Angluin’ s learning framework