Representing Regular Languages of Infinite Words Using Mod 2 Multiplicity Automata

Dana Angluin[1], Timos Antonopoulos[1], Dana Fisman[2], and Nevin George[1]

[1] Yale University, New Haven, CT, USA
[2] Ben-Gurion University, Beer-Sheva, Israel

Abstract. We explore the suitability of mod 2 multiplicity automata (M2MAs) as a representation for regular languages of infinite words. M2MAs are a deterministic representation that is known to be learnable in polynomial time with membership and equivalence queries, in contrast to many other representations. Another advantage of M2MAs compared to non-deterministic automata is that their equivalence can be decided in polynomial time and complementation incurs only an additive constant size increase. Because learning time is parameterized by the size of the representation, particular attention is focused on the relative succinctness of alternate representations, in particular, LTL formulas and Büchi automata of the types: deterministic, non-deterministic and strongly unambiguous. We supplement the theoretical results of worst case upper and lower bounds with experimental results computed for randomly generated automata and specific families of LTL formulas.

1 Introduction

Regular languages of infinite words (or ω-words) play an important role in verification of reactive systems. The question of whether a system S satisfies a specification given by a temporal logic formula φ can be reduced to the question of whether $L(S) \cap L(\neg\varphi)$ is empty, where $L(S)$ is the set of ω-words representing the computation paths of the system S and $L(\neg\varphi)$ is the set of ω-words representing computations that violate φ. Automata are a useful machinery for performing operations on languages such as complementation and intersection, and for deciding properties such as emptiness and equivalence. Many verification tools are implemented using reductions to automata [20].

Regular ω-languages can be represented using various types of automata (e.g. Büchi, Rabin, Parity, etc.). Different automata types differ in their succinctness and in the complexity of performing operations of interest. Non-deterministic Büchi automata (NBAs) are one of the most popular acceptor types for regular ω-languages, mainly due to their simplicity, succinctness, and good complexity for the emptiness problem. An issue with Büchi automata is that their deterministic version (DBAs) is strictly less expressive: while NBAs accept all regular ω-languages, DBAs recognize only a strict subset thereof. Another issue is that complementation of NBAs is hard; it has a $2^{\Omega(n \log n)}$ lower bound (where n is the number of states) [16]. This motivated the introduction of *complete unambiguous*

Büchi automata (CUBA) by Carton and Michel who showed that every regular ω-language can be represented by a CUBA, i.e. there is a way to limit the non-determinism without losing expressiveness [8]. Bousquet and Löding proposed *strongly unambiguous Büchi automata* (SUBA), a slight relaxation of CUBA for which they have shown that equivalence can be decided in polynomial time [6].

The SUBA model was also shown useful in terms of learnability of regular ω-languages — Angluin, Antonopoulos and Fisman have shown that SUBAs are polynomially predictable using membership queries (while NBAs, under plausible cryptographic assumptions, are not) [1]. Their proof makes use of a model of automata called *Mod 2 Multiplicity Automata* (M2MA). Informally, *multiplicity automata* are an algebraic variant of automata that compute functions from finite words to a field $\mathcal{K}$ [4,5], and *M2MAs* are multiplicity automata that work over the field $GF(2) = \{0, 1\}$ where sum and product are computed modulo 2.

In this paper we look at questions concerning the adequacy of M2MAs for representing regular ω-languages. We note that M2MAs operate on finite words, and their use for representing regular ω-languages follows a reduction, by Calbrix, Nivat and Podelski from a regular ω-language L to a regular language $(L)_\$$ of finite words [7]. We thus start by reviewing the succinctness of M2MAs with respect to automata on finite words, particularly of types non-deterministic (NFAs), deterministic (DFAs), and unambiguous (UFAs). We show that M2MAs are more succinct than DFAs and UFAs, whereas with respect to NFAs there are in the worst case exponential gaps in going from M2MAs to NFAs and vice versa.

We also study the complexity of performing basic operations on M2MAs; complementation can be done with an additive constant increase in size, and union and intersection with the product of sizes. There is a known cubic algorithm to minimize a weighted automaton [10], which applies to an M2MA and also implies cubic procedures for determining emptiness and equivalence.

We then investigate the succinctness of M2MAs in representing regular ω-languages, by comparing translations from *linear temporal logic* (LTL) formulas and Büchi automata (deterministic, non-deterministic and strongly unambiguous) into M2MAs, DFAs, UFAs, SUBAs and NBAs (where the former three use the $(L)_\$$ representation). The results are summarized in Fig. 3.

To complement the theoretical bounds, we implemented procedures to transform SUBAs to UFAs and UFAs to M2MAs, and to minimize and learn M2MAs, and report estimates of the average size increases in transforming random SUBAs, DBAs, and NBAs to M2MAs. We also determine the minimum dimensions of M2MAs and minimum sizes of DFAs for a few members of three specific families of LTL formulas and compare them with the respective ω-automaton sizes.

2 Preliminaries

For nonnegative integers k and ℓ, $[k..\ell]$ is the set of nonnegative integers n such that $k \leq n \leq \ell$. Given a finite alphabet Σ, Σ^* is the set of finite words over Σ. The length of a word x is $|x|$ and the empty word is ε. $\Sigma^n = \{x \in \Sigma^* \mid |x| = n\}$. The reverse of a word x is x^r. A language L is any subset of Σ^*. The reverse of

L, denoted L^r, is $\{x^r \mid x \in L\}$. The *Hankel matrix* of a language L is the infinite matrix whose rows and columns are indexed by elements of Σ^*, where the entry for row x and column y is 1 if $xy \in L$ and 0 if $xy \notin L$.

The set of infinite words (or ω-words) over Σ is the set of all maps from the positive integers to Σ and is denoted Σ^ω. An ω-language is any subset of Σ^ω. For a finite or infinite word w, $w[i]$ denotes the symbol at position i, with indices starting at 1. Concatenation of a finite word x with a finite or infinite word y is denoted xy. The word x is a prefix of xy and the word y is a suffix of xy. The suffix of w starting at position i is denoted $w[i:]$. If $x \in \Sigma^*$ and k is a nonnegative integer, x^k denotes the concatenation of k copies of x, and x^ω denotes the concatenation of x with itself infinitely many times. An ω-word is ultimately periodic if it can be written in the form $u(v)^\omega$ for $u, v \in \Sigma^*$ with $|v| > 0$. If A_1 and A_2 are sets and $S \subseteq A_1 \times A_2$, then we define the projection $\pi_1(S) = \{a_1 \mid (\exists a_2)(a_1, a_2) \in S\}$ and analogously for the projection π_2.

2.1 NFAs, UFAs, DFAs, NBAs, UBAs, SUBAs, and DBAs

A (nondeterministic) finite-state automaton A is a tuple $(\Sigma, Q, I, \Delta, F)$ consisting of a finite alphabet Σ, a finite set Q of states, a set $I \subseteq Q$ of initial states, a set $F \subseteq Q$ of final states, and a transition relation $\Delta \subseteq Q \times \Sigma \times Q$. The transition relation Δ is deterministic if for every state $q \in Q$ and every symbol $\sigma \in \Sigma$, there is at most one state $q' \in Q$ such that $(q, \sigma, q') \in \Delta$. The size of a finite-state automaton is $|Q|$.

For a word w, a run of A on w is a sequence of states $q_0, q_1, \ldots$ such that for each i that indexes a symbol in w, $(q_{i-1}, w[i], q_i) \in \Delta$. Thus, for $w \in \Sigma^*$ a run on w is a sequence of length $|w| + 1$, and for $w \in \Sigma^\omega$, a run on w is an infinite sequence of states. A run on w is *initial* if $q_0 \in I$. A finite run is *final* if $q_{|w|} \in F$, and an infinite run is *final* if there are infinitely many values of i for which $q_i \in F$. Acceptors of languages and ω-languages may be defined using finite-state automata, as follows. In each case, the language of words accepted by an acceptor A is denoted $L(A)$.

A *nondeterministic finite acceptor* (NFA) is a finite-state automaton A that accepts a word $w \in \Sigma^*$ if there exists a run of A on w that is both initial and final. An NFA A is an *unambiguous finite acceptor* (UFA) if for every word $w \in L(A)$ there is exactly one run of A on w that is initial and final. An NFA A is a *deterministic finite acceptor* (DFA) if there is exactly one initial state ($|I| = 1$) and the transition relation Δ is deterministic. The languages over Σ that are accepted by NFAs, UFAs, or DFAs is precisely the regular languages over Σ.

A *nondeterministic Büchi acceptor* (NBA) is a finite-state automaton A that accepts a word $w \in \Sigma^\omega$ if there exists a run of A on w that is both initial and final. An NBA is an *unambiguous Büchi acceptor* (UBA) if for every $w \in L(A)$, there exists exactly one run of A on w that is initial and final. Bousquet and Löding [6] introduced the concept of a *strongly unambiguous Büchi acceptor* (SUBA), which is an NBA such that for every $w \in \Sigma^\omega$, there is at most one final run of the acceptor on w — note that the condition of being initial is dropped.

Thus, every SUBA is a UBA. The ω-languages over Σ that are accepted by NBAs, UBAs, or SUBAs are precisely the regular ω-languages. An NBA is a *deterministic Büchi acceptor* (DBA) if there is exactly one initial state ($|I| = 1$) and the transition relation Δ is deterministic. Every DBA is a UBA, but is not necessarily a SUBA. The ω-languages that are accepted by DBAs are a proper subclass of the class of all regular ω-languages.

For Büchi acceptors, we also consider a generalized version, GNBA, in which the acceptance condition is specified not by a single set of final states, but by a collection $\mathcal{F}$ of sets of final states. For a GNBA, a run $q_0, q_1, \ldots$ is *final* iff for each $F \in \mathcal{F}$, there exist infinitely many indices i such that $q_i \in F$. Applying this generalization to a SUBA yields a GSUBA. There is a standard translation of a GNBA of size n with k sets of final states into an NBA of size kn, in which there are k copies of the GNBA automaton. However, applying this construction to a GSUBA does not in general yield a SUBA.

2.2 LTL formulas

The syntax of *linear temporal logic* (LTL) [18] over a set AP of atomic propositions is given by the following grammar $\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid (\varphi_1 \,\mathcal{U}\, \varphi_2)$ where $p \in AP$ is an atomic proposition.

The semantics of LTL relates ω-words over 2^{AP} to formulas as shown on the right (recall that indexing of words starts at 1). Additional Boolean and temporal connectives are defined in the usual way. In particular $\top$ (*true*) is defined as $p \vee \neg p$, $\Diamond\varphi$ (*eventually* φ) is defined as $(\top \,\mathcal{U}\, \varphi)$ and $\Box\varphi$ (*always* φ) is defined as $\neg\Diamond(\neg\varphi)$.

$w \models p$	iff	$p \in w[1]$
$w \models \neg\varphi$	iff	$w \not\models \varphi$
$w \models \varphi_1 \wedge \varphi_2$	iff	$w \models \varphi_1$ and $w \models \varphi_2$
$w \models \bigcirc\varphi$	iff	$w[2:] \models \varphi$
$w \models (\varphi_1 \,\mathcal{U}\, \varphi_2)$	iff	$\exists j.\ w[j:] \models \varphi_2$ and $\forall i < j.\ w[i:] \models \varphi_1$

The ω-language of an LTL formula φ, denoted $L(\varphi)$, is the set of ω-words for which it is true. The *size* of an LTL formula φ is the number of distinct subformulas it contains. Every LTL formula represents a regular ω-language (see Section 5). However, not every regular ω-language can be represented by an LTL formula; in particular, the regular ω-languages that can be represented by LTL formulas are noncounting [9].

2.3 M2MAs

A *multiplicity automaton* represents a function mapping Σ^* to elements of a field $\mathcal{K}$. We focus on the case where $\mathcal{K} = \{0, 1\}$ and product and sum are computed modulo 2. A *mod 2 multiplicity acceptor* (M2MA) of dimension d is a tuple $A = (\Sigma, v_I, \{\mu_\sigma\}_{\sigma\in\Sigma}, v_F)$, where Σ is the input alphabet, $v_I \in \mathcal{K}^d$ is the initial vector, $v_F \in \mathcal{K}^d$ is the final vector, and for each $\sigma \in \Sigma$, μ_σ is a $d \times d$ transition matrix over $\mathcal{K}$, that is, an element of $\mathcal{K}^{d\times d}$.

The vectors v_I and v_F are interpreted as $d \times 1$ column vectors. The transpose operation is denoted by $^\top$, and the inner product of two column vectors $v, w \in \mathcal{K}^d$ is denoted $v^\top w$.

To define $L(A)$ we inductively define the matrix μ_x for all $x \in \Sigma^*$. If $x = \varepsilon$, then μ_x is the $d \times d$ identity matrix. If $x = \sigma y$ for some $\sigma \in \Sigma$ and $y \in \Sigma^*$ then $\mu_x = \mu_\sigma \mu_y$. The function $f_A : \Sigma^* \to \mathcal{K}$ computed by A is defined by $f_A(x) = v_I^\top \mu_x v_F$. A word x is *accepted* by A if $f_A(x) = 1$.

We refer to column vectors $v \in \mathcal{K}^d$ as *states* or *co-states* of A. A state v is *reachable* iff there exists a word $x \in \Sigma^*$ such that $v = (v_I^\top \mu_x)^\top$. A co-state w is *co-reachable* iff there exists a word $x \in \Sigma^*$ such that $w = \mu_x v_F$. For any state v, $L_v(A)$ denotes the language of words accepted by A with its initial vector replaced by v.

We assume standard results from finite dimensional vector spaces. If U is a vector space of dimension k over the field $\{0, 1\}$ then $|U| = 2^k$. If U is a vector subspace of the vector space V, then the *orthogonal complement* of U is the set $U^\perp = \{v \mid v^\top u = 0 \ \forall u \in U\}$, $U^\perp$ is a vector subspace of V which is disjoint from U except for the zero vector, and the dimensions of U and $U^\perp$ sum to the dimension of V.

The following simple lemmas relate M2MAs to UFAs and DFAs, and show that M2MAs accept exactly the regular languages.

Lemma 1. *[Beimel et al. [4]] Let $L \subseteq \Sigma^*$. If L is accepted by a UFA of size n, it is also accepted by an M2MA of dimension n.*

Lemma 2. *Let $L \subseteq \Sigma^*$. If L is accepted by an M2MA of dimension d with R reachable states, then L is also accepted by a DFA of R states. Clearly, $R \leq 2^d$.*

Beimel et al. [4] have shown that there is a polynomial time algorithm to learn an unknown M2MA using equivalence and membership queries.

2.4 Size lower bounds for DFAs, M2MAs and NFAs

Given a language $L \subseteq \Sigma^*$, we define an *observation table* for L as an $\ell \times m$ matrix T of 0's and 1's where each row i is associated with a finite word x_i and each column j is associated with a finite word y_j, and the entry $T_{i,j}$ is 1 if and only if $x_i y_j \in L$. This terminology is derived from its use in algorithms to learn DFAs. An observation table for L is thus a finite submatrix of its Hankel matrix.

Certain properties of observation tables for a language L yield lower bounds on acceptors recognizing L. Recall that the rank of a matrix is the number of linearly independent rows (or columns) it contains.

Lemma 3. *Let T be an observation table for the regular language L with rows associated with finite words x_i for $i = [1..\ell]$ and columns associated with finite words y_j for $j \in [1..m]$. Assume T has n distinct rows and rank d over the field $\{0, 1\}$. Then any DFA to accept L must have at least n states, and any M2MA to accept L must have dimension at least d.*

Proof. Let D be a DFA accepting L. If the rows for x_i and x_k are distinct, then there is a column j on which they differ, that is, $x_i y_j \in L$ iff $x_k y_j \notin L$. Thus, the states of D reached from the initial state on the words x_i and x_k must be different and D has at least n states.

Let M be an M2MA accepting L. Following the argument of Beimel et al. [4], the observation table is a submatrix of the Hankel matrix of the language L, and its rank (modulo 2) is a lower bound for the rank (modulo 2) of the Hankel matrix, which is a lower bound for the size of any M2MA accepting L. □

For lower bounds for NFAs, we use the concept of covering the observation table by 1-monochromatic rectangles. If R and C are subsets of the indices of the rows and columns (respectively) of a matrix M, then the (R, C)-*rectangle* of M is the matrix obtained from M by deleting those rows whose indices are not in R and those columns whose indices are not in C. The (R, C)-rectangle of a matrix M is v-*monochromatic* iff all of its entries are equal to the value v.

Let M be a matrix of 0 and 1 values. A 1-*rectangle cover* of M is a set $\{(R_s, C_s) \mid s \in [1..t]\}$, of 1-monochromatic rectangles (R_s, C_s) of M such that for every i and j, if $M_{i,j} = 1$ then there exists some $s \in [1..t]$ such that $i \in R_s$ and $j \in C_s$. A *minimum* 1-*rectangle cover* of M is a 1-rectangle cover of M of minimum possible cardinality t.

Lemma 4. *Let T be an $\ell \times m$ observation table for the regular language L. Any NFA M recognizing L must have at least as many states as the cardinality of the minimum* 1*-rectangle cover of T.*

This is implied by Theorem 5.2.4.10 and Exercise 5.2.5.14 of Hromkovič [12]. For completeness we provide a simple direct proof.

Proof. Let the strings indexing the rows of T be x_i for $i \in [1..\ell]$ and the strings indexing the columns of T be y_j for $j \in [1..m]$. For each state q of M, let R_q be the set of all $i \in [1..\ell]$ such that x_i reaches q from an initial state of M, and let C_q be the set of all $j \in [1..m]$ such that y_j reaches a final state of M from q.

Clearly (R_q, C_q) must be a 1-monochromatic rectangle of T, because if $i \in R_q$ and $j \in C_q$ then $x_i y_j$ is accepted by M and the entry $T_{i,j}$ must be 1. Also, if $T_{i,j} = 1$, then $x_i y_j$ must be accepted by M, so there must exist a state q of M such that x_i reaches q from an initial state of M and y_j reaches a final state of M from q, that is, $i \in R_q$ and $j \in C_q$. Thus, the rectangles (R_q, C_q) for all states q of M form a 1-rectangle covering of T, and the number of states of M is greater than or equal to the cardinality of the minimum 1-rectangle covering of T. □

Corollary 5. *If L is a regular language with an $n \times n$ observation table T that has exactly one* 1 *in every row and column, then any DFA, M2MA, or NFA to recognize L must have at least n states.*

As an example of the use of these results, let L be the regular language over $\{a, b, c\}$ consisting of those strings that do not contain any occurrences of the substrings ba or cb, with the observation table for L in Fig. 1. There are 4 different rows, so any DFA to accept L must have at least 4 states. The mod 2 rank of the table is 3 (the first three rows are a row basis) so any M2MA accepting L must have dimension

	ε	a	b
ε	1	1	1
b	1	0	1
c	1	1	0
ba	0	0	0

Fig. 1: Observation table with rank 3.

at least 3. The observation table with rows c and b, and columns a and b is the 2×2 identity matrix, so any NFA to accept L must have at least 2 states. In fact, there is a DFA of 4 states, an M2MA of dimension 3, and an NFA of 2 states accepting L, so for this example, the lower bounds are tight.

3 M2MAs as representations of regular languages

We consider the computational cost and size implications of some common operations and decision questions using M2MAs to represent regular languages.

3.1 M2MAs: procedures for operations and properties

Reverse Given an M2MA A accepting a regular language L, an M2MA A^r accepting the reverse language L^r may be obtained from A by exchanging the initial and final vectors, and transposing each of the transition matrices. Thus, the minimum dimension of an M2MA accepting L is equal to the minimum dimension of an M2MA accepting L^r. Reversing is similarly easy for UFAs and NFAs, but may incur an exponential increase in size for a DFA.

Sum If for $i = 1, 2$, M_i is a multiplicity automaton of dimension d_i computing the function $f_i : \Sigma^* \to \mathcal{K}$, then the sum $f_1 + f_2$ is computed by a multiplicity automaton M of dimension $d_1 + d_2$ constructed as the direct product of M_1 and M_2 as follows. State vectors of M are the concatenation of state vectors of M_1 and M_2, including the initial and final vectors. For each $\sigma \in \Sigma$, the transition matrix μ_σ is a $(d_1+d_2) \times (d_1+d_2)$ matrix obtained by putting $(\mu_1)_\sigma$ in the upper left, $(\mu_2)_\sigma$ in the lower right, and setting the remaining entries to 0. This ensures that the state updates of M_1 and M_2 are done in parallel for each symbol, and the output is the sum of the outputs for M_1 and M_2.

Boolean operations For M2MAs, complementation follows directly from the sum construction. If A is an M2MA of dimension d and C is the M2MA of dimension 1 that outputs 1 on every string, then the sum construction with M and C yields an M2MA of dimension $d+1$ that accepts the regular language $\Sigma^* \setminus L(A)$. For DFAs, complementation is size-preserving, while for NFAs, complementation may incur an exponential increase in size.

Given M2MAs A_i of dimension d_i for $i = 1, 2$, the intersection language $L(A_1) \cap L(A_2)$ is accepted by an M2MA of dimension $d_1 \cdot d_2$ obtained from A_1 and A_2 using the Kronecker product of matrices.[3] Union can then be obtained from complementation and intersection.

[3] If A is an $m \times n$ matrix and B is a $p \times q$ matrix, then the Kronecker product $A \otimes B$ is the $pm \times qn$ block matrix, with blocks of size B, where the block-matrix at position (i, j) is $a_{ij}B$ [17, Def 1.2.1].

Minimization, Equivalence, and Emptiness Sakarovitch [10] describes a cubic-time algorithm to minimize a weighted automaton with weights from a skew field, which has the following corollary.

Corollary 6 (of Theorem 5.20 in [10]). *Given an M2MA A of dimension d, an M2MA A' of the minimum possible dimension accepting $L(A)$ may be found in time $O(|\Sigma|d^3)$.*

An M2MA recognizes the empty language iff it has dimension 0 when minimized, and the equivalence of two M2MAs may be tested by determining if their sum is the empty language. Pseudocode for M2MA minimization is in Appendix C.

3.2 Conciseness comparisons for regular languages

We summarize known results comparing the conciseness of M2MAs with that of DFAs, UFAs and NFAs as representations of regular languages in Fig. 2. The entry for row A and column B is "$-$" if the representation A is an instance of the representation B, otherwise, starting with a machine of size n in the representation A, how large must an equivalent machine in the representation B be in the worst case? The entry $2^{\Theta(n)}$ means that there is a lower bound of 2^{cn} and an upper bound of 2^{dn} for positive constants c and d.

We briefly explain the entries in the table. A DFA is also a UFA and an NFA, and a UFA is also an NFA. A DFA or UFA of size n can be converted to an equivalent M2MA of dimension n (Lemma 1). The subset construction to determinize an NFA of size n yields a DFA (and therefore also a UFA or M2MA) of size at most 2^n. An M2MA of dimension n can be converted to a DFA (or UFA or NFA) of size at most 2^n (Lemma 2). The language $B_n = \Sigma^* \cdot 1 \cdot \Sigma^n$, for $\Sigma = \{0, 1\}$, consisting of binary strings with a 1 located $n + 1$ symbols before the end is accepted by a UFA of size $n + 2$ (and therefore also an NFA of size $n + 2$ and an M2MA of dimension $n + 2$), but requires at least 2^{n+1} states for any DFA that accepts it.

	DFA	UFA	NFA	M2MA
DFA	$-$	$-$	$-$	n
UFA	$2^{\Theta(n)}$	$-$	$-$	n
NFA	$2^{\Theta(n)}$	$2^{\Theta(n)}$	$-$	$2^{\Theta(n)}$
M2MA	$2^{\Theta(n)}$	$2^{\Theta(n)}$	$2^{\Theta(n)}$	$-$

Fig. 2: Worst-case size bounds for representations of regular languages.

For the problem of converting an NFA to an M2MA, Kaznatcheev and Panangaden [13] consider the language $L_n = \Sigma^* \left((0\Sigma^{n-1}1) + (1\Sigma^{n-1}0)\right) \Sigma^*$ for $\Sigma = \{0, 1\}$, and show that L_n is recognized by an NFA of $2n+2$ states, but that any M2MA to recognize L_n must have dimension at least 2^n. By Lemma 1, this lower bound applies also to UFAs.

For the problem of converting an M2MA to an NFA, Kaznatcheev and Panangaden [13] give a family of languages $\{L_n\}$ such that L_n is recognized by an M2MA of dimension $n + 2$, and prove that any NFA to recognize L_n must have at least $2^{n/2} - 2$ states. Here we provide a simpler proof of a stronger lower bound. Let L_n be the language recognized by the M2MA given in Fig. 1 of the paper of Kaznatcheev and Panangaden. This M2MA accepts a word iff the number of indices i such that both $w[i]$ and $w[i + n]$ is 1, is odd.

Lemma 7. *Any NFA to recognize L_n must have at least 2^{n-1} states.*

Proof. The language L_n has an observation table T_n of dimension $2^n \times 2^n$, in which the rows and columns are indexed by strings $x, y \in \{0,1\}^n$. We view strings in $\{0,1\}^n$ as vectors of length n over the field $\{0,1\}$, so that the entry corresponding to the pair (x, y) is the inner product of the vectors x and y, that is $x^\top y$. Note that the inner product $x^\top y$ is 1 iff the number of indices i such that both $xy[i]$ and $xy[i+n]$ is 1, is odd. The lower bound of $2^n - 1$ then follows from Lemma 4 and the following Lemma. □

Lemma 8. *The minimum 1-rectangle covering of the observation table T_n just defined has cardinality $2^n - 1$.*

Proof. For the upper bound it suffices to consider a 1-rectangle covering of T_n consisting of pairs (R, C) where R is the singleton index of a nonzero row and C consists of the indices of the occurrences of 1 in that row.

If $x \in \{0,1\}^n$ is the zero vector, then $x^\top y$ is 0 for all vectors y; otherwise, $x^\top y = 1$ for exactly half the vectors y, that is, for 2^{n-1} columns of T_n. Hence, T_n contains exactly $2^{n-1}(2^n - 1)$ entries of value 1. We now show that any 1-monochromatic rectangle (R, C) of T_n has at most 2^{n-1} entries of 1, which shows that a minimum 1-rectangle covering of T_n must have cardinality at least $2^n - 1$.

Let (R, C) be any 1-monochromatic rectangle of T_n. Let U be the vector subspace spanned by the vectors x corresponding to indices in R, and let B be a basis for U whose indices are drawn from R. Let $k = |B|$, so that $|U| = 2^k$. Every element of U is a sum of elements of B, but a sum of an even number of elements of B will be 0 in all the columns with indices in C, so R can contain the indices of at most half the elements of U, that is, $|R| \leq 2^{k-1}$.

Let $S = \{v \mid u^\top v = 1 \ \forall u \in B\}$, the set of vectors whose inner product with all elements of B is 1; clearly, $|C| \leq |S|$. We use inclusion/exclusion to find the cardinality of S, as follows.

$$\begin{aligned}
|S| &= 2^n - |\bigcup_{u \in B} \{v \mid u^\top v = 0\}| \\
&= 2^n - |\bigcup_{C \subseteq B} C^\perp| \\
&= 2^n - k2^{n-1} + \binom{k}{2} 2^{n-2} - \ldots (-1)^k 2^{n-k} \\
&= 2^n \cdot (1 - \frac{1}{2})^k \\
&= 2^{n-k}
\end{aligned}$$

Thus, $|C| \leq 2^{n-k}$. Then $|R \times C| \leq 2^{k-1} \cdot 2^{n-k} = 2^{n-1}$, concluding the proof. □

4 Representing regular omega-languages using regular languages

In the preliminaries we discussed NBAs, SUBAs and DBAs, and LTL formulas as representations of regular ω-languages. Here we explain that M2MAs and other automata over finite words can also be used to represent regular ω-languages.

A regular ω-language is uniquely determined by the set of ultimately periodic ω-words it contains. Let L be a regular ω-language and let \$ be a symbol not in the alphabet of L. To represent the set of ultimately periodic words in L, Calbrix, Nivat and Podelski [7] introduced the related language of finite words $L_{\$} = \{u\$v \mid u(v)^{\omega} \in L\}$ and proved that it is regular.

Thus a regular ω-language L can be represented by an acceptor for the regular language $L_{\$}$, for example, a DFA, UFA, NFA or M2MA. The representation of $L_{\$}$ by an M2MA was used by Angluin, Antonopoulos, and Fisman [1] in showing that regular ω-languages are polynomially predictable with membership queries as a function of the size of the smallest SUBA accepting the language.

We note that if for $i = 1, 2$, A_i is an M2MA of dimension d_i accepting $(L_i)_{\$}$ for the regular ω-language L_i, then there is an M2MA of dimension $d_1 \cdot d_2$ accepting $(L_1 \cap L_2)_{\$}$, and an M2MA of dimension $d_1 + 3$ accepting $(\Sigma^{\omega} \setminus L_1)_{\$}$. The former follows by the intersection result for M2MAs, and the latter follows by the sum result applied to A_1 and the dimension 3 M2MA that accepts the set $\{u\$v \mid u \in \Sigma^*, v \in \Sigma^+\}$.

5 Conciseness comparisons for regular omega-languages

We present known and new results comparing the conciseness of M2MAs with that of several other representations of regular ω-languages, summarized in Fig. 3. The entry for row A and column B gives upper (above) and lower (below) bounds on the worst case increase in size for a representation of type A of size or dimension n to an equivalent representation of type B. The entry is "$-$" if a representation of type A is an instance of a representation of type B. The entries for the columns for DFA, UFA, M2MA, and NFA are for the language $L_{\$}$. An arrow indicates that the (lower or upper) bound is derived from a related (lower or upper) bound in the table. For example, the upper bound for the row DBA and columns UFA, M2MA and NFA are derived from the upper bound for the row DBA and column DFA. We now discuss the entries.

5.1 Size increases for LTL formulas

Upper bounds
There is a "classic" algorithm, described by Baier and Katoen [3, Chapter 5], to translate an LTL formula of size n into a GNBA of size 2^n with at most n sets of final states, which then yields an NBA of size at most $n2^n$. This shows that every LTL formula represents a regular ω-language, and gives an upper bound for translating an LTL formula to an NBA. Another algorithm to translate LTL formulas into NBAs is given by Gerth, Peled, Vardi and Wolper [11].

	DFA	UFA	M2MA	NFA	(G)SUBA	NBA
LTL	$2^{2^{O(n)}}$ via UFA	$2^{O(n)}$ Cor.11	$\leftarrow$	$\leftarrow$	$(2^n, n)$ Prop. 9	$n2^n$ [3]
	$\rightarrow$	$\rightarrow$	$2^{\Omega(n)}$ Thm. 14	$2^{\Omega(n)}$ Thm. 14	$\rightarrow$	$2^{\Omega(n)}$ [3]
DBA	$n + n3^{n^2}$ [14]	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\downarrow$	–
	$2^{\Omega(n \log n)}$ [2]	$\rightarrow$	$2^{\Omega(n)}$ Thm. 15	$2^{\Omega(n)}$ Thm. 15	$2^{\Omega(n)}$ [6]	–
NBA	$2^n + 2^n3^{n^2}$ [14]	$\leftarrow$	$\leftarrow$	$n + n3^{n^2}$ [14]	$(12n)^n$ [8]	–
	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	$\uparrow$	–
SUBA	$\uparrow$	$2n^2 + n$ [6]	$\leftarrow$	$\leftarrow$	–	–
	$2^{\Omega(n)}$ [1]	$\rightarrow$	$2n^2 - n + 2$ Thm. 16	$2n^2 - n + 2$ Thm. 16	–	–

Fig. 3: Worst-case size bounds for representations of regular ω-languages.

Concerning the classic translation algorithm, Bousquet and Löding [6] give a brief argument and state that "Hence the automaton that is constructed in this standard way is strongly unambiguous." Wilke [21] states that "Every temporal formula with n subformulas can be translated into an equivalent backwards deterministic generalized Büchi automaton with at most 2^n states and as many Büchi sets as there are subformulas with leading temporal operator F (eventually) or U (until)." To clarify these earlier statements, we reformulate them in our terminology. This gives an upper bound for transforming an LTL formula to a GSUBA.

Proposition 9. *Let ϕ be an LTL formula of size n with temporal operators next and until, with m until subformulas. Applying the classic translation algorithm to ϕ yields a GSUBA of size 2^n with m sets of final states.*

Proof. Baier and Katoen [3] show that the algorithm yields a GNBA M of the given size in which each state corresponds to an assignment of true or false to every subformula of ϕ. Moreover, if the ω-word w is accepted from a state q, then q assigns true to each subformula ψ of ϕ iff ψ is true for w. Hence there is at most one state of M from which the ω-word w is accepted, and thus M is also GSUBA. $\square$

To get an upper bound for translation of LTL formulas to UFAs, M2MAs, and NFAs, we would like to use the property of being strongly unambiguous. However, if the resulting GSUBA has more than one set of final states, transforming it in the usual way into an NBA does not in general yield a SUBA. Instead, we generalize to GSUBAs the method of Bousquet and Löding [6] for transforming a SUBA accepting L into a UFA accepting $L_{\$}$.

Theorem 10. *There is an algorithm to transform a GSUBA of size n with m sets of final states accepting L into a UFA of size $2^m n^2 + n$ accepting $L_\$$. It runs in time polynomial in n and 2^m.*

Proof. Let L be accepted by the GSUBA $M = (\Sigma, Q, I, \Delta, \mathcal{F})$ with $n = |Q|$ and $m = |\mathcal{F}|$. We index the elements of $\mathcal{F}$ as F_i for $i \in [1..m]$. Bousquet and Löding [6, Lemma 1] show that $u(v)^\omega$ is accepted by a SUBA iff there exists a state q reachable from an initial state on reading u, such that on the word v there is a computation path that loops from q back to q while passing through an accepting state. For the GSUBA M, the condition is that the computation path that loops from q back to q must pass through at least one state from each F_i for $i \in [1..m]$.

We define an NFA $M' = (\Sigma', Q', I', \Delta', F')$ as follows. The alphabet is $\Sigma' = \Sigma \cup \{\$\}$. The state set is $Q' = Q \cup Q_1$, where $Q_1 = \{(q_1, q_2, S) \mid q_1, q_2 \in Q, S \subseteq [1..m]\}$. The initial states are $I' = I$. The transition relation is $\Delta' = \Delta \cup \Delta_1 \cup \Delta_2$, where Δ_1 is the set of all triples $((q_1, q_2, S), \sigma, (q_1', q_2', S'))$ such that $q_1' = q_1$, $(q_2, \sigma, q_2') \in \Delta$, and $S' = S \cup T$, where $T = \{i \in [1..m] \mid q_2' \in F_i\}$. And Δ_2 contains all triples $(q, \$, (q, q, \emptyset))$ such that $q \in Q$. The set of final states F' is the set of triples (q_1, q_2, S) such that $S = [1..m]$ and $q_1 = q_2$.

Then M' has $2^m n^2 + n$ states, and can be constructed in time polynomial in n and 2^m given the GSUBA M. On an input $u\$v$, the NFA M' behaves like M on the word u, reaching some state q. Then on the symbol \$, M' transitions to the state $(q, q, \emptyset)$, recording the state q reached after reading u. As M' continues reading v, the first component remembers q while the second component transitions as in M. The third component, S, records the set of indices of those final sets F_i that have been visited in the processing of v. The input $u\$v$ is accepted by M' iff there is a state q of M reachable from a state of I on input u such that there exists a computation path in M on input v from q to q that visits at least one state in F_i for every $i \in [1..m]$. Thus M' accepts $L_\$$. (Note that the set S generalizes the single bit used in Bousquet and Löding's construction.)

To see that M' is a UFA, we note that if there are two different accepting computations in M' for $u\$v$, then these may be used to construct two different accepting computations in M for $u(v)^\omega$, contradicting the fact that M is a GSUBA. □

The entry in Fig. 3 for row LTL and column UFA is then justified by the following.

Corollary 11. *Let ϕ be an LTL formula of size n with temporal operators next and until, with m until subformulas. Then there is a UFA of size $2^{2n+m} + 2^n$ to accept $L(\phi)_\$$.*

For transforming LTL to DFA, we have only the doubly-exponential bound for transforming an LTL formula to a UFA and the UFA to DFA.

Lower bounds

We first generalize Lemma 3 to DBAs and Lemma 4 to NBAs. An *observation* *table* for an ω-language L is a matrix $T \in \{0,1\}^{\ell \times m}$ with rows indexed by finite words x_i for $i \in [1..\ell]$ and columns indexed by ω-words y_j for $j \in [1..m]$ such that $T_{i,j} = 1$ iff $x_i y_j \in L$. Then we have the following, proved analogously to Lemma 3 and Lemma 4.

Lemma 12. *Let T be an observation table for the ω-language L. If T has n distinct rows, then any DBA accepting L has at least n states.*

Lemma 13. *Let T be an observation table for the ω-language L. If the minimum* 1*-cover of T has cardinality n, then any NBA to recognize L has at least n states.*

Baier and Katoen [3, Theorem 5.4.2] give a lower bound for a family of LTL formulas ϕ_n of size poly(n) for which equivalent NBAs must have at least 2^n states. Below we give a simplified and slightly strengthened version of their lower bound, which also applies to M2MAs or NFAs for $L_\$$.

Theorem 14. *For every positive integer n there exists an LTL formula ψ_n of size at most $2n+6$ such that any NBA accepting $L(\psi_n)$ must have size at least 2^n. Any NFA or M2MA accepting $L(\psi_n)_\$$ must have size or dimension at least 2^n.*

Proof. Let p be a propositional variable. For any positive integer n we define the LTL formula $\psi_n = \Box(p \to \bigcirc^n(p)) \wedge (\neg p \to \bigcirc^n(\neg p))$. We use $\bigcirc^n$ to represent the composition of $\bigcirc$ with itself n times, so $\bigcirc^3(p)$ abbreviates $\bigcirc(\bigcirc(\bigcirc(p)))$. The formula ψ_n has size $2n+6$. Let the symbols 0 and 1 represent the assignment of false and true to p. Then $L(\psi_n)$ is the language of ω-words w over $\{0,1\}$ such that for some $x \in \Sigma^n$, $w = x^\omega$.

For $L(\psi_n)_\$$, let $x_1, x_2, \ldots, x_{2^n}$ be any total ordering of all the elements of $\{0,1\}^n$, and consider the observation table T with rows corresponding to x_i and columns corresponding to $\$x_i$ for $i \in [1..2^n]$. Clearly, there is exactly one 1 in row x_i, in the column $\$x_i$, so this observation table is the $2^n \times 2^n$ identity matrix, which has rank 2^n, and any NFA or M2MA accepting $L(\psi_n)_\$$ must have size at least 2^n by Corollary 5.

For the lower bound on NBAs, we observe that if we instead index the columns of T with $(x_i)^\omega$, it becomes an observation table for the ω-language $L(\psi_n)$, and remains the $2^n \times 2^n$ identity matrix, which implies that any NBA accepting $L(\phi_n)$ must have at least 2^n states, by Lemma 13. $\square$

5.2 Size increases for DBAs, NBAs, SUBAs

Upper bounds

For an NBA of n states accepting L, Calbrix, Nivat and Podelski [7] show that there is a DFA of $2^n + 2^{2n^2+n}$ states to accept $L_\$$. Kuperberg, Pinault and Pous [14] give a more concise construction that yields for $L_\$$ an NFA of size $n + n3^{n^2}$ and a DFA of size $2^n + 2^n 3^{n^2}$. For the conversion of an NBA of n states to a SUBA, Carton and Michel provide the upper bound of $(12n)^n$ [8].

Starting with a DBA instead of an NBA, the NFA construction of Kuperberg, Pinault and Pous is fully deterministic, so the upper bound of $n+n3^{n^2}$ holds for transforming a DBA into a DFA. Bousquet and Löding [6] show that a SUBA of n states accepting the ω-language L may be transformed into a UFA of $2n^2+n$ states accepting $L_\$$.

Lower bounds

For transforming a DBA for L into a DFA for $L_\$$, Angluin and Fisman [2] prove that for every n there is a DBA of $n+2$ states accepting a language L such that no DFA of fewer than $n!$ states accepts $L_\$$. For transforming a DBA into a UFA, M2MA or NFA, we prove the following result.

Theorem 15. *For every even positive integer n there is an ω-language L_n that is accepted by a DBA of $n+5$ states such that any UFA, NFA or M2MA to accept $(L_n)_\$$ must have size or dimension at least $\binom{n}{n/2}$, which is $\sim 2^n/\sqrt{\pi n/2}$.*

Proof (Sketch). The proof, given in full in Section A.1, uses a modification of the DBAs in the construction by Angluin and Fisman [2]. Here we sketch the main idea and give an example. Let $n=2k$ for some nonnegative integer k, let $\Sigma_{2k}=\{\sigma_1,\ldots,\sigma_{2k}\}$ and let Σ be $\Sigma_{2k}\cup\{0,L,E,F\}$. Consider the regular ω-language defined by the ω-regular expression $\left(\cup_{\sigma\in\Sigma\setminus\{0\}}\ (\sigma\cdot(\Sigma\setminus\{\sigma\})^*\cdot\sigma)\right)^\omega$, which is accepted by a DBA with $2k+5$ states. Given two subsets C and D of Σ_{2k}, each of size k, we define words u_C and v_D such that $(u_C\cdot v_D)^\omega$ is in the language if and only if $C=D$. The main idea behind the construction is that v_D forces each symbol σ_D in $\Sigma_{2k}\setminus D$ to be followed by the character 0. Thus, if the string preceding (and including) an occurrence of such a symbol σ_D is described by the (unambiguous) regular expression $(\bigcup_{\sigma\in\Sigma\setminus\{0\}}\sigma\cdot(\Sigma\setminus\{\sigma\})^*\cdot\sigma)^*$, then the symbol 0 that follows cannot be properly consumed, resulting in the ω-word being not in the language. We construct the words u_C and v_D in such a way that this can happen if and only if such a symbol $\sigma_D\in\Sigma_{2k}\setminus D$ is also in C. Since C and D are subsets of Σ_{2k}, each of size k, this happens exactly when $C\neq D$. There is therefore an observation table with rows indexed by $\$u_C$ for all subsets C of size k and whose columns are indexed by v_D for all subsets D of size k, and where each entry, corresponding to row and column subsets C and D respectively, is 1 if and only if $C=D$. By Corollary 5, the result follows. □

Example. Let $\Sigma_{2k}=\{1,2,3,4\}$, let Σ be $\Sigma_{2k}\cup\{0,L,E,F\}$, let $C=\{2,3\}$, and let $D=\{2,4\}$. Then u_C,v_C and v_D are defined on the right. Then $(u_C\cdot v_C)^\omega$ is in the language, whereas $(u_C\cdot v_D)^\omega$ is not (since $C\neq D$).

$$u_C=F\cdot 2\cdot F\cdot 2\cdot 3\cdot 2\cdot 3\cdot L\cdot 3$$
$$v_C=L\cdot E\cdot 1\cdot 0\cdot 4\cdot 0\cdot E$$
$$v_D=L\cdot E\cdot 1\cdot 0\cdot 3\cdot 0\cdot E$$

For the lower bound on transforming a DBA into a SUBA, Bousquet and Löding [6] show that for every positive integer n there exists an ω-language that is accepted by a DBA with $n+1$ states, and cannot be accepted by a SUBA with fewer than 2^{n-1} states.

For transforming a SUBA into a DFA, Angluin, Antonopoulos and Fisman [1, Theorem 5] give a family of ω-languages such that L_n is accepted by a SUBA of size $4n+5$, but any DFA to accept $(L_n)_\$$ or its reverse must have size at least 2^n. For transforming a SUBA into a UFA, M2MA or NFA, we prove the following asymptotically tight lower bound.

Theorem 16. *For every positive integer m greater than 3, there is an ω-language L that is accepted by a SUBA with m states, but no M2MA of dimension less than $2m^2 - m + 2$ or NFA or UFA of size less than $2m^2 - m + 2$ accepts $(L)_\$$.*

Proof (Sketch). For every $n \in \mathbb{N}$ we define L_n to be the regular ω-language over $\Sigma = \{a, b, c\}$ given by the expression $((cc \cdot b^n)^* \cdot aa \cdot b^n)^\omega$. This language is accepted by the SUBA S_n, with $m = n + 3$ states, shown in Fig. 8 in Appendix A.2. We construct a specific observation table M for the language $(L_n)_\$$. We then show that any 1-rectangle cover of M is of size at least $2m^2 - m + 2$, which implies by Lemma 4 that the number of states of any NFA (or UFA) for the language $(L_n)_\$$ is at least $2m^2 - m + 2$. We further show that the rank of M is $2m^2 - m + 2$, and by Lemma 3, obtain that the dimension of any M2MA for this language is also at least $2m^2 - m + 2$. □

6 Empirical results

We report typical size increases in going from a random SUBA, DBA or NBA acceptor for a regular ω-language L to a minimized M2MA (and DFA, in the case of a SUBA) for $L_\$$. We also report computed sizes of minimized M2MAs and DFAs for $L(\phi_n)_\$$ for members of particular families $\{\phi_n\}$ of LTL formulas. Code is available in the GitHub repository:
https://github.com/nevingeorge/Learning_Automata.

For the generation of random SUBAs, DBAs or NBAs, our procedure is as follows. Given parameters n, f, and t we generate a transition relation on n states (random reverse-deterministic for a SUBA, random deterministic for a DBA, and all possible transitions for an NBA), select f of the n states at random to be final, and randomly remove t of the transitions. The resulting transition relation is trimmed to remove non-live states and their transitions. The trimmed acceptor may have fewer than n states.

If the goal is a SUBA, using the criterion of Wilke [21], we check that there do not exist two different states q_1 and q_2 and a nonempty finite word v such that for $i = 1, 2$, there is a loop on v from q_i to q_i that passes through a final state. If the acceptor fails this test, it is rejected, and the procedure is repeated until a SUBA is successfully generated.

6.1 SUBAs to minimized M2MAs and DFAs

For random SUBAs to minimized M2MAs, we first generate a random SUBA with $\Sigma = \{a, b, c\}$, $n \in \{5, 10, 15\}$, $t \in \{[1, 5], [2, 10], [18, 22]\}$ (resp.), and $f = 2$ or $f = 3$ with equal probability. We then convert it into a UFA using the algorithm of Bousquet and Löding [6], and minimize the equivalent M2MA.

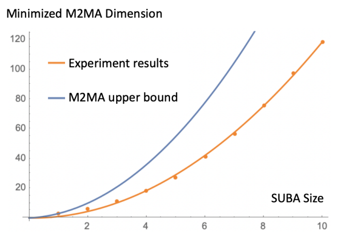


Fig. 4: Random SUBAs to minimized M2MAs

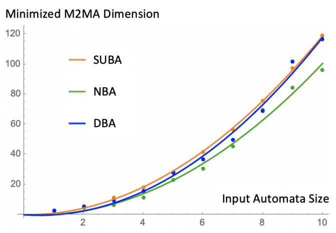
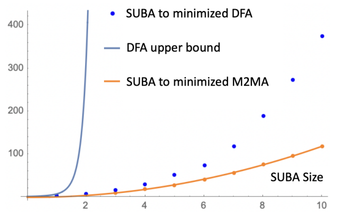


Fig. 5: Random SUBAs, NBAs, and DBAs to minimized M2MAs



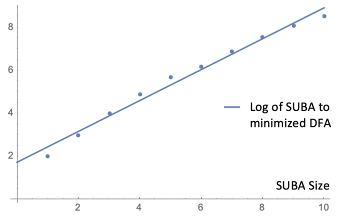


Fig. 6: Random SUBAs to minimized DFAs

We performed the above process on approximately $220,000$ randomly generated SUBAs.

Fig. 4 is a plot of the average minimized M2MA dimension for each trimmed SUBA size from 1 to 10. Upon performing quadratic regression, we obtain the orange curve $1.212n^2 - .2248n$, and the blue curve is the theoretical upper bound of $2n^2 + n$ given in Fig. 3. The quadratic fit has a R^2 of 0.9996 while a linear fit has a R^2 of 0.9370, suggesting that the growth is indeed quadratic. This curve satisfies the theoretical upper bound of $2n^2 + n$, and suggests that the lower bound of $\Omega(n^2)$ holds on average.

For random SUBAs to minimized DFAs, we also calculated the number of reachable states of each minimized M2MA. This is the number of states in the equivalent minimized DFA, by a property of the minimization algorithm of Corollary 6. From Fig. 3, the lower bound in going from a SUBA to a DFA is $2^{\Omega(n)}$, and the upper bound is $2^n + 2^n 3^{n^2}$.

In the left graph in Fig. 6, the blue data points representing the results of the SUBA to DFA experiment grow much more sharply than the results of the SUBA to M2MA experiment, so it is clear that a SUBA can be represented more concisely as an M2MA than as a DFA on average. Upon taking the log (base 2), we obtain a roughly linear fit as seen in the right graph with equation $.7196n + 1.738$ and a R^2 of .9841, suggesting that on average the growth is exponential. As shown in Fig. 10 in Appendix B.1, the standard deviation and range of converted DFA sizes was large for this conversion, making it difficult to make firm claims about the growth. However, the data suggests that the exponential lower bound likely holds on average, and that in general the upper bound of $2^n + 2^n 3^{n^2}$ is a severe overestimate.

6.2 NBAs and DBAs to minimized M2MAs

For NBAs and DBAs, a minimized M2MA is computed using the M2MA learning algorithm of Beimel et al. [4], which makes membership and equivalence queries to the NBA or DBA. Instead of exact equivalence queries, we use approximate equivalence queries, implemented by testing membership agreement on a sample of randomly generated ultimately periodic words. Thus, the dimension of the learned M2MA may be an underestimate of the true minimum dimension of an M2MA for $L_{\$}$.

For the NBA/DBA to M2MA experiments, we generated approximately 1000 random NBAs/DBAs with $\Sigma = \{a, b, c\}$, $n \in \{5, \ldots, 10\}$, $t \in [0, n]$ for DBAs and t in ranges within $[90, 680]$ for NBAs (see exact details in Fig. 13 of Appendix B.1), and $f = 2$ or $f = 3$ with equal probability. For the approximate equivalence queries, we tested 1000 random ultimately periodic words of length at most 25. The results of the experiments can be seen in Fig. 5. The fitted NBA and DBA curves are quadratic with equations $1.096n^2 - .8947n$ and $1.318n^2 - 1.392n$, respectively. The quadratic fits for the NBA and DBA results have a R^2 of .9954 and .9961, respectively, while linear fits have a R^2 of .9227 and .9118, respectively. These experiments have limitations: the use of approximate equivalence queries, the small sample size (because of the time requirements of the learning algorithm), and the large standard deviation and range of converted M2MA sizes as shown in Fig. 11 and Fig. 12 in Appendix B.1. However, the results from all three conversions are very similar, suggesting that in these conditions, SUBAs, NBAs, and DBAs don't vary significantly on average with respect to their equivalent M2MA representations.

6.3 LTL formulas to minimized M2MAs

Random LTL formulas seem not to provide much insight, so we consider specific families of LTL formulas: bounded request/grant formulas and two families based on the hierarchy of Manna and Pnueli [15], namely obligation and reactivity formulas. Empirically, for each of the first few members of each family we calculate the minimum dimension of an M2MA and the minimum size of a DFA accepting the corresponding $L_{\$}$ language, and use the online tool provided by the Spot website (https://spot.lrde.epita.fr/) to find an ω-language acceptor for the corresponding L. (Omitted Spot entries exceeded the limit on calculation time.)

The canonical request/grant formula is of the form $\Box(p \rightarrow \Diamond(q))$, which asserts that whenever a request (p) is made, it is eventually granted (q). In the bounded version, a number of steps n is specified, and the assertion is that the request is granted within n steps. Thus, for each natural number n, we have a formula $R_n = \Box(p \rightarrow (q \vee \bigcirc(q) \vee \bigcirc^2(q) \vee \ldots \vee \bigcirc^n(q)))$. The table in Fig. 7a gives the resulting sizes and dimensions for n from 0 to 5. It is reasonable to conjecture $n + 1$ for the size of a DBA, $n^2 + 3n + 3$ for the minimum dimension of an M2MA, and $2n^2 + 3n + 4$ for the minimum size of a DFA representing R_n.

n	DBA	M2MA	DFA
0	1	3	4
1	2	7	9
2	3	13	18
3	4	21	31
4	5	31	48
5	6	43	69

(a) R_n sizes.

n	DBA	M2MA	DFA
1	3	7	9
2	9	19	23
3	27	55	63
4	81	163	179
5	–	487	519

(b) F_n sizes.

n	GNBA	M2MA	DFA
1	(4,1)	5	6
2	$(10, 2)$	11	12
3	$(28, 3)$	29	30
4	–	83	84
5	–	245	246

(c) G_n sizes.

Fig. 7: Size or dimension of acceptors for families of LTL formulas.

The family of obligation formulas we consider is: $F_n = \wedge_{i=1}^{n}(\Box p_i \vee \Diamond q_i)$. Using conjunction and minimization, we calculate the minimum dimension M2MA (and minimum size DFA) for $L_\$$ for these formulas for n up to 5. The table in Fig. 7b shows the results. It is reasonable to conjecture 3^n for the size of a DBA, $2 \cdot 3^n + 1$ for the minimum dimension of an M2MA, and $2 \cdot 3^n + 2^n + 1$ for the minimum size of a DFA to represent F_n.

The family of reactivity formulas we consider is: $G_n = \wedge_{i=1}^{n}(\Box\Diamond p_i \vee \Diamond\Box q_i)$. We proceed as for the obligation formulas, with the results shown in the table in Fig. 7c. Note that these formulas cannot be represented by DBAs, but are instead represented by GNBAs, which may have multiple sets of final states. For example, the entry $(10, 2)$ indicates a GNBA with 10 states and 2 sets of final states. A reasonable conjecture in this case is $(3^n + 1, n)$ for the size of a GNBA, $3^n + 2$ for the minimum dimension of an M2MA, and $3^n + 3$ for the minimum size of a DFA representing G_n.

In these cases, the minimum dimension of an M2MA (and size of a DFA) appears to grow at most as a polynomial in the size of an ω-language acceptor, quadratically for the bounded request/grant family, and linearly for the obligation and reactivity families.

7 Summary and conclusions

We provide a survey of size relations of M2MAs as a representation of regular languages and regular ω-languages, as well as empirical results for several of these relations. New theoretical results include an improvement of the lower bound for transforming an M2MA to an NFA, an upper bound of $2^{O(n)}$ for the translation of an LTL formula of size n to a UFA, NFA, or M2MA, a lower bound of $2^{\Omega(n)}$ for the translation of a DBA of n states to an M2MA or NFA, and an asymptotically optimal lower bound of $2n^2 - n + 2$ for the translation of a SUBA of n states to an M2MA or NFA.

M2MAs have many advantages as a representation for regular ω-languages: determinism, succinct complementation, and polynomial time algorithms for minimization, equivalence testing, and learning with membership and equivalence queries. M2MAs are as succinct as DFAs, sometimes exponentially more so, and deserve further study.

References

1. D. Angluin, T. Antonopoulos, and D. Fisman. Strongly unambiguous Büchi automata are polynomially predictable with membership queries. In *28th EACSL Annual Conference on Computer Science Logic, CSL*, pages 8:1–8:17, 2020.
2. D. Angluin and D. Fisman. Learning regular omega languages. *Theor. Comput. Sci.*, 650:57–72, 2016.
3. C. Baier and J-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.
4. A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, May 2000.
5. F. Bergadano and S. Varricchio. Learning behaviors of automata from multiplicity and equivalence queries. *SIAM J. Comput.*, 25(6):1268–1280, 1996.
6. N. Bousquet and C. Löding. Equivalence and inclusion problem for strongly unambiguous Büchi automata. In *Language and Automata Theory and Applications, 4th International Conference, LATA. Proceedings*, pages 118–129, 2010. `doi:10.1007/978-3-642-13089-2\_10`.
7. H. Calbrix, M. Nivat, and A. Podelski. Ultimately periodic words of rational w-languages. In *Proceedings of the 9th International Conference on Mathematical Foundations of Programming Semantics*, pages 554–566. Springer-Verlag, 1994.
8. O. Carton and M. Michel. Unambiguous Büchi automata. *Theor. Comput. Sci.*, 297(1-3):37–81, 2003. `doi:10.1016/S0304-3975(02)00618-7`.
9. V. Diekert and P. Gastin. First-order definable languages. In *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas].*, pages 261–306, 2008.
10. Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*, chapter 4: Rational and Recognizable Series, by Jaques Sakarovitch, pages 105–174. Springer-Verlag Berlin Heidelberg, 2009.
11. R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV. PSTV 1995.* Springer, 1996. `doi:https://doi.org/10.1007/978-0-387-34892-6\_1`.
12. Juraj Hromkovič. *Communication Complexity and Parallel Computing.* Springer-Verlag Berlin Heidelberg, 1997. (There is also 2013 edition.).
13. Artem Kaznatcheev and Prakash Panangaden. Weighted automata are compact and actively learnable. *Information Processing Letters*, 171, 2021. (The authors were apparently unaware of prior results on learning multiplicity automata by Beimel et al. and others.).
14. D. Kuperberg, L. Pinault, and D. Pous. Coinductive algorithms for Büchi automata. In *Developments in Language Theory - 23rd International Conference, DLT Proceedings*, pages 206–220, 2019.
15. Z. Manna and A. Pnueli. A hierarchy of temporal properties (invited paper, 1989). In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, PODC '90, page 377–410. Association for Computing Machinery, 1990. `doi:10.1145/93385.93442`.
16. M. Michel. Complementation is much more difficult with automata on infinite words. In *Manuscript, CNET*, 1988.
17. Barry Kurt Moser. Linear algebra and related introductory topics. In *Linear Models, A Mean Model Approach, A volume in Probability and Mathematical Statistics*, pages 1–22, 1996.
18. A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

19. M. R. Thon and H. Jaeger. Links between multiplicity automata, observable operator models and predictive state representations: a unified learning framework. *Journal of Machine Learning Research*, 16:103–147, 2015.
20. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency - Structure versus Automata (8th Banff Higher Order Workshop, Banff, Canada, August 27 - September 3, 1995, Proceedings)*, pages 238–266, 1995. `doi:10.1007/3-540-60915-6\_6`.
21. T. Wilke. ω-automata. *CoRR*, abs/1609.03062, 2016. URL: `http://arxiv.org/abs/1609.03062`, `arXiv:1609.03062`.

A Appendix: Deferred proofs

Deferred proofs are given here.

A.1 Lower bound for transforming a DBA to an M2MA or NFA

Theorem 15 (restated) *For every even positive integer n there is an ω-language L_n that is accepted by a DBA of $n+5$ states such that any UFA, NFA or M2MA to accept $(L_n)_\$$ must have size or dimension at least $\binom{n}{n/2}$, which is $\sim 2^n/\sqrt{\pi n/2}$.*

Proof. Let $n = 2k$ for some nonnegative integer k, and let $\Sigma_{2k} = \{\sigma_1, \ldots, \sigma_{2k}\}$, be an ordered alphabet (with $\sigma_i < \sigma_{i+1}$ for all $i < 2k$). Let Σ be $\Sigma_{2k} \cup \{0, L, E, F\}$. For each combination C from Σ_{2k} of size k, define the strings u_C and v_C as follows. Let $C = \{\sigma_{i_1}, \ldots, \sigma_{i_k}\} \subseteq \Sigma_{2k}$, and let the elements in C be such that for all $j \in [1, .., k-1]$, $\sigma_{i_j} < \sigma_{i_{j+1}}$. For any i_j define $C\downharpoonright_{i_j}$ to be the subset of C containing all elements larger than or equal to σ_{i_j}. Then we define the string $u_{C\downharpoonright_{i_j}}$ inductively as follows:

$$u_{C\downharpoonright_{i_j}} \begin{cases} \sigma_{i_j} \cdot L \cdot \sigma_{i_j} & \text{if } |C\downharpoonright_{i_j}| = 1, \\ \sigma_{i_j} \cdot \sigma_{i_{j+1}} \cdot \sigma_{i_j} \cdot u_{C\downharpoonright_{i_{j+1}}} & \text{if } |C\downharpoonright_{i_j}| > 1. \end{cases}$$

Then u_C is simply defined to be $F \cdot \sigma_{i_1} \cdot F \cdot u_{C\downharpoonright_{i_1}}$.

Example 17. Suppose that $\Sigma_{2k} = \{1, 2, 3, 4\}$. Then for $C = \{2, 4\}$, u_C is defined to be $F2F2424L4$ and for $C' = \{1, 2\}$, $u_{C'}$ is $F1F1212L2$.

For $C = \{\sigma_{i_1}, \ldots, \sigma_{i_k}\} \subseteq \Sigma_{2k}$, let $\overline{C} = \{\sigma_{z_1}, \ldots, \sigma_{z_k}\}$ be $\Sigma_{2k} \setminus C$. Notice that both C and $\overline{C}$ have size k, since $|\Sigma_{2k}| = 2k$. Then let x_C be the string $\sigma_{z_1} \cdot 0 \cdots \sigma_{z_k} \cdot 0$. Finally, we define v_C to be equal to $L \cdot E \cdot x_C \cdot E$.

Example 18. Suppose that $\Sigma_{2k} = \{1, 2, 3, 4\}$. Then for $C = \{2, 4\}$, v_C is defined to be $LE1030E$ and for $C' = \{1, 2\}$, $v_{C'}$ is $LE3040E$.

Let L be the following language defined in [2]. Let $A = (\Sigma, Q, Q_I, \Delta, Q_F)$ be the deterministic Büchi automaton, where $\Sigma = \{0, \ldots, 2k, E, F, L\}$, $Q = \{q_0, q_1, \ldots, q_{2k}, q_E, q_L, q_F, q_\perp\}$, with initial set of states Q_I and final set of states Q_F equal to $\{q_0\}$, and Δ contains exactly the tuples below:

$$\begin{aligned} (q_0, \sigma, q_\sigma) &\in \Delta \quad \text{for all } \sigma \in \Sigma \setminus \{0\}, \\ (q_0, 0, q_\perp) &\in \Delta, \\ (q_\sigma, \sigma', q_\sigma) &\in \Delta \quad \text{for all } \sigma \in \Sigma \setminus \{0\} \text{ and } \sigma' \in \Sigma \setminus \{\sigma\}, \\ (q_\sigma, \sigma, q_0) &\in \Delta \quad \text{for all } \sigma \in \Sigma \setminus \{0\}, \\ (q_\perp, \sigma, q_\perp) &\in \Delta \quad \text{for all } \sigma \in \Sigma. \end{aligned}$$

Let L be the language accepted by A. Notice that the regular expression

$$\left(\bigcup_{\sigma \in \Sigma \setminus \{0\}} \sigma \cdot (\Sigma \setminus \{\sigma\})^* \cdot \sigma \right)^\omega$$

defines the same language.

Claim. Let $\mathcal{C}$ be the set of all combinations of elements of Σ_{2k} of size k, where $|\Sigma_{2k}| = 2k$. Then for any $C \in \mathcal{C}$, $(u_C \cdot v_C)^\omega$ is in L, whereas for any $C, C' \in \mathcal{C}$ with $C \neq C'$, the string $(u_C \cdot v_{C'})^\omega$ is not in L.

Proof of Claim Let A be the deterministic Büchi automaton accepting the language L as defined above. Since Δ is deterministic, for each $\sigma \in \Sigma$ and $q \in Q$, there is at most one q' such that $(q, \sigma, q') \in \Delta$. We define the function δ from $Q \times \Sigma^*$ to Q, to be the one that maps (q, s) to q', if q' is the unique state reached on input s when starting from state q.

Let $C \in \mathcal{C}$ be equal to $\{\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_k}\}$ with $\sigma_{i_j} < \sigma_{i_{j+1}}$ for all $j < k$. By definition of the string u_C, the following holds for the mapping δ:

$$\begin{aligned}
\delta(q_0, u_C) &= q_0 \\
\delta(q_L, u_C) &= q_{i_k} \\
\delta(q_F, u_C) &= q_L \\
\delta(q_{i_1}, u_C) &= q_F \\
\delta(q_{i_j}, u_C) &= q_{i_{j-1}} \text{ for all } 1 < j \leq k \\
\delta(q_z, u_C) &= q_z \text{ for all } z \text{ s.t. } \sigma_z \in \overline{C}.
\end{aligned}$$

We omit the case when we start with state q_E, for readability, as it is not needed in this argument. Similarly, regarding the string v_C, the following holds for the mapping δ:

$$\begin{aligned}
\delta(q_0, v_C) &= q_L \\
\delta(q_L, v_C) &= q_0 \\
\delta(q_F, v_C) &= q_F \\
\delta(q_{i_j}, v_C) &= q_{i_j} \text{ for all } 1 \leq j \leq k, \\
\delta(q_z, v_C) &= q_\perp \text{ for all } z \text{ s.t. } \sigma_z \in \overline{C}.
\end{aligned}$$

Then, by composing the above we obtain:

$$\begin{aligned}
\delta(q_0, u_C \cdot v_C) &= q_L \\
\delta(q_L, u_C \cdot v_C) &= q_{i_k} \\
\delta(q_F, u_C \cdot v_C) &= q_0 \\
\delta(q_{i_1}, u_C \cdot v_C) &= q_F \\
\delta(q_{i_j}, u_C \cdot v_C) &= q_{i_{j-1}} \text{ for all } 1 < j \leq k, \\
\delta(q_z, u_C \cdot v_C) &= q_\perp \text{ for all } z \text{ s.t. } \sigma_z \in \overline{C}.
\end{aligned}$$

Thus, it holds that $\delta(q_0, (u_C \cdot v_C)^2) = q_{i_k}$, $\delta(q_{i_k}, (u_C \cdot v_C)^{k-1}) = q_{i_1}$ and $\delta(q_{i_1}, (u_C \cdot v_C)^2) = q_0$. These imply that $\delta(q_0, (u_C \cdot v_C)^{k+3}) = q_0$ and therefore, $(u_C \cdot v_C)^\omega$ is in L.

On the other hand, suppose $C' \in \mathcal{C}$ is such that $C \neq C'$, and let ℓ be the largest value less than or equal to k, such that $\sigma_{i_\ell} \in C \cap \overline{C'}$. Considering the strings u_C and $v_{C'}$, we have:

$$\begin{aligned}
\delta(q_0, u_C \cdot v_{C'}) &= q_L \\
\delta(q_L, u_C \cdot v_{C'}) &= \begin{cases} q_{i_k} & \text{if } \ell \neq k, \\ q_\bot & \text{if } \ell = k \end{cases} \\
\delta(q_F, u_C \cdot v_{C'}) &= q_0 \\
\delta(q_{i_1}, u_C \cdot v_{C'}) &= q_F, \\
\delta(q_{i_j}, u_C \cdot v_{C'}) &= \left\{ \begin{array}{ll} q_{i_{j-1}} & \text{if } \ell \neq j-1, \\ q_\bot & \text{if } \ell = j-1 \end{array} \right\} \text{ when } 1 < j \leq k, \\
\delta(q_z, u_C \cdot v_{C'}) &= q_\bot \text{ for all } z \text{ s.t. } \sigma_z \in \overline{C}.
\end{aligned}$$

It follows that $\delta(q_0, (u_C \cdot v_{C'})^{k-\ell+2}) = q_\bot$ and therefore $(u_C \cdot v_{C'})^\omega$ is not in L. This completes the proof of the claim.

From the claim it follows that the observation matrix indexed by the rows $\$u_C$ and columns $v_{C'}$, for $C, C' \in \mathcal{C}$, is the identity matrix (i.e. entry is 1 when $C = C'$ and 0 otherwise), and thus of rank $|\mathcal{C}|$. The size of $\mathcal{C}$ is equal to $\binom{2k}{k}$, which is $\sim 2^{2k}/\sqrt{\pi \cdot k}$, which is greater than or equal to 2^k. On the other hand, the DBA that accepts L has $2k+5$ states. The result follows by Corollary 5. □

A.2 Lower bound for transforming a SUBA to an M2MA or NFA

Theorem 16 (restated) *For every positive integer m greater than 3, there is an ω-language L that is accepted by a SUBA with m states, but no M2MA of dimension less than $2m^2 - m + 2$ or NFA or UFA of size less than $2m^2 - m + 2$ accepts $(L)_\$$.*

Proof. For $n \in \mathbb{N}$, let L_n be the regular ω-language over $\Sigma = \{a, b, c\}$ defined by the expression $((cc \cdot b^n)^* \cdot aa \cdot b^n)^\omega$. This language is accepted by the SUBA S_n shown in Fig. 8, with set of states $Q = \{q_0, q_a, q_c, q_1, \ldots, q_n\}$ (of size $n+3$).

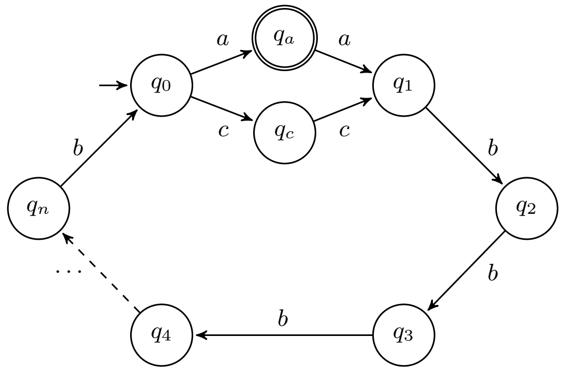


Fig. 8

Consider the $(L_n)_\$$ language for L_n, defined as $\{u \cdot \$ \cdot v \mid u \cdot v^\omega \in L_n\}$. We describe the construction of two observation tables, M_1 and M_2, for $(L_n)_\$$, which are combined to form the observation table M, which witnesses the lower bounds. (There is, in fact, a DFA of $2|Q| - |Q| + 2$ states to recognize $(L_n)_\$$, so the lower bounds match the upper bounds in this case.)

For each $q \in Q$, we define s_q to be a string of minimal length that can reach state q from the initial state q_0. Let I_+ be the set $\{(q, q', +) \mid q, q' \in Q\}$ and let I_- be the set $\{(q, q', -) \mid q, q' \in Q \setminus \{q_a\}\} \cup \{(q_a, q_a, -)\}$. The size of I_+ is $|Q|^2$, the size of I_- is $(|Q| - 1)^2 + 1$, and the size of their union is $2 \cdot |Q|^2 - 2 \cdot |Q| + 2$.

For each $(q, q', +) \in I_+$, we define $e_{(q,q',+)}$ to be the shortest string of length at least two and with at least one 'a' or at least one 'c' that can reach state q' from state q by visiting q_a. Similarly, for each tuple $(q, q', -) \in I_-$, different from $(q_a, q_a, -)$, we define $e_{(q,q',-)}$ to be the shortest string of length at least two and with at least one 'a' or at least one 'c', that can reach state q' from state q without visiting q_a. Finally, we define $e_{(q_a,q_a,-)}$ to be the empty string, ε. Denote the set of all these strings by E, and note that they are all pairwise distinct.

For the first observation table, M_1, let the rows be labeled s_q for $q \in Q$, and let the columns be labeled $\$e_{(q,q,+)}$. M_1 has dimension $|Q| \times |Q|$ and exactly one 1 in every row and column, because $s_q\$e_{(q',q',+)}$ is in $(L_n)_\$$ iff $q = q'$.

We construct the second observation table M_2 as follows. For each tuple $(q, q', +) \in I_+$, define $x_{(q,q',+)}$ to be $s_q \cdot \$ \cdot e_{(q,q',+)}$, and for each tuple $(q, q', -) \in I_-$, define $x_{(q,q',-)}$ to be $s_q \cdot \$ \cdot e_{(q,q',-)}$. Let X denote the set of all those strings, and note that $|X| = |Q|^2 - 2 \cdot |Q| + 2$.

Let M_2 be the observation table for $(L_n)_\$$ indexed by rows X and columns E, such that an entry for row x and column e is 1 if and only if $x \cdot e \in (L_n)_\$$. The observation table M_2 may be described as follows.

- For each pair $(q, q') \in Q \times Q$, such that $q \neq q_a$, $q' \neq q_a$, and $q \neq q'$, there are exactly two 1's in the row $x_{(q,q',+)}$, in the columns $e_{(q',q,+)}$ and $e_{(q',q,-)}$, and there is exactly one 1 in the row $x_{(q,q',-)}$, in the column $e_{(q',q,+)}$.
- For each pair (q, q) with $q \in Q$ and $q \neq q_a$, there are exactly three 1's in the row $x_{(q,q,+)}$, in the columns $e_{(q,q,+)}$, $e_{(q,q,-)}$, and in the column $e_{(q_a,q_a,-)}$, because $x_{(q,q,+)} \in (L_n)_\$$. However, there is exactly one 1 in the row $x_{(q,q,-)}$, in the column $e_{(q,q,+)}$.
- For each pair $(q, q') \in Q \times Q$ with exactly one of q, q' being equal to q_a (*i.e.* either $q = q_a$ and $q' \neq q_a$ or $q \neq q_a$ and $q' = q_a$), there is exactly one 1 in row $x_{(q,q',+)}$, in the column $e_{(q',q,+)}$.
- Finally, for (q_a, q_a), there are two 1's in the row $x_{(q_a,q_a,+)}$, in the columns $e_{(q_a,q_a,+)}$ and $e_{(q_a,q_a,-)}$ and one 1 in the row $x_{(q_a,q_a,-)}$, in the column $e_{(q_a,q_a,+)}$.

The combined observation table M has all the row and column labels of M_1 and M_2. Because none of the row labels of M_1 contain the symbol \$ and all of the row labels of M_2 contain the symbol \$, we may visualize the table M as a copy of M_1 in the dimension $|Q| \times |Q|$ upper left corner, and a copy of M_2 in the dimension $2|Q| - 2|Q| + 2$ lower right corner, and 0's elsewhere.

Lower bound for NFAs: No 1-rectangle in M may cover 1's in both the M_1 and M_2 portions of M, so we may argue about them separately. The M_1

portion has one 1 in every row and column, and therefore requires $|Q|$ many 1-rectangles to cover. We show that any 1-rectangle covering of M_2 must have at least $2|Q|^2 - 2|Q| + 2$ rectangles. For any $q, q' \in Q \setminus \{q_a\}$, let $M_{(q,q')}$ be the submatrix of M_2 with rows $\{x_{(q,q',+)}, x_{(q,q',-)}\}$ and columns $\{e_{(q',q,+)}, e_{(q',q,-)}\}$. Then, according to the analysis of M_2, any 1-rectangle cover of the submatrix $M_{(q,q')}$ will need two 1-monochromatic rectangles. These entries will require $2(|Q|-1)^2$ many 1-monochromatic rectangles in a 1-rectangle cover of the matrix M_2.

Let X_{q_a} be the set of $x_{(q,q',+)}$ where either q or q' is equal to q_a, but not both of them are, and similarly, let E_{q_a} be the set of $e_{(q,q',+)}$ where either q or q' is equal to q_a, but not both of them are. Let M_{q_a} be the submatrix of M_2 indexed by the rows in X_{q_a} and columns in E_{q_a}. This submatrix is an $\ell \times \ell$ identity matrix, where ℓ is $2 \cdot (|Q| - 1)$, and requires $2|Q| - 2$ many 1-monochromatic rectangles in a 1-rectangle cover of the matrix M_2.

Finally, the submatrix of rows $x_{(q_a,q_a,+)}$ and $x_{(q_a,q_a,-)}$ and columns $e_{(q_a,q_a,+)}$ and $e_{(q_a,q_a,-)}$ has two 1's in its first row and one 1 in its second row, and requires two more 1-monochromatic rectangles in a 1-rectangle cover of the matrix M.

These submatrix covers can be part of the 1-rectangle covering for the whole matrix M_2, but importantly, the rectangles involved in these coverings cannot be extended to include all the remaining entries of M_2. Therefore, any 1-rectangle covering for M_2 will contain at least $2\cdot(|Q|-1)^2+2\cdot(|Q|-1)+2 = 2|Q|^2-2|Q|+2$ many elements. Combining the results for M_1 and M_2, any 1-rectangle covering of M must contain at least $|Q|+2|Q|^2-2|Q|+2 = 2|Q|^2-|Q|+2$ many elements. This is a lower bound on the number of states of any NFA recognizing $(L_n)_\$$ by Lemma 4.

Lower bound for M2MAs: We claim that the rank of the matrix M is also $2|Q|^2 - |Q| + 2$. We describe row operations (adding one row of the matrix to another row) that convert it into a matrix M' with one 1 in every row and column, showing that M has rank equal to its dimension. The M_1 portion of M already satisfies this condition, so we consider the M_2 portion.

First, the only difference between rows $x_{(q_a,q_a,+)}$ and $x_{(q_a,q_a,-)}$ is in column $e_{(q_a,q_a,-)}$. In particular, $x_{(q_a,q_a,+)} \in (L_n)_\$$, but $x_{(q_a,q_a,-)} \notin (L_n)_\$$. Thus, adding the latter row to the former yields a row with a single 1 in column $e_{(q_a,q_a,-)}$, which may then be added to each row $x_{(q,q,+)}$ with $q \neq q_a$ to cancel the (third) 1 in that column.

After this operation, every row and column has one or two 1's, and the only occurrences of two 1's in a row or column occur in the pairs of rows $x_{(q,q',+)}$ and $x_{(q,q',-)}$ and columns $e_{(q',q,+)}$ and $e_{(q',q,-)}$ where $q \neq q_a$ and $q' \neq q_a$. Recall that such a submatrix has two 1's in its first row and one 1 in its second row, so adding the second row to the first will transform the two rows and columns so that each now has exactly one 1. Thus, the rank of M is $2|Q|^2-|Q|+2$. This is a lower bound on the dimension of any M2MA recognizing $(L_n)_\$$ by Lemma 3. □

B Appendix: Empirical results

B.1 Tables of results for transforming SUBAs, NBAs, and DBAs

We include more detailed information about size increases for random SUBAs, NBAs, and DBAs in Figs. 9, 10, 11, 12, and 13.

SUBA Size	Mean	Median	Std Dev	Min	Max	Count
1	3.00	3.00	0.00	3	3	53803
2	6.09	7.00	1.52	4	8	35247
3	11.37	13.00	3.74	4	17	36751
4	18.50	21.00	7.05	4	30	39925
5	27.55	31.33	10.88	4	47	29313
6	41.55	44.50	17.44	7	68	5279
7	56.86	60.00	23.84	8	93	6189
8	75.81	80.00	31.31	8	122	6238
9	97.33	101.00	37.00	9	155	4915
10	119.03	125.00	45.04	11	192	2364

Fig. 9: SUBA to minimized M2MA

SUBA Size	Mean	Median	Std Dev	Min	Max	Count
1	4.00	4.00	0.00	4	4	53803
2	8.05	9.00	2.31	5	14	35247
3	16.12	16.33	7.26	5	58	36751
4	30.20	28.33	18.76	5	237	39925
5	52.48	47.67	36.56	6	422	29313
6	73.94	61.00	63.17	2	2015	5279
7	119.18	95.50	122.11	2	5607	6189
8	188.55	147.00	187.55	2	6467	6238
9	273.74	208.50	345.55	2	18964	4915
10	374.41	283.00	350.06	2	5101	2364

Fig. 10: SUBA to minimized DFA

C Appendix: M2MA minimization algorithm

Because the algorithm given by Sakarovitch [10] that proves Corollary 6 is described in somewhat abstract terms, we provide a more concrete description and pseudocode for it. We note that the minimization algorithm for weighted automata given by Thon and Jaeger [19] requires the Moore-Penrose inverse, and therefore does not apply to an M2MA.

Let $A = (\Sigma, v_I, \{\mu_\sigma\}_{\sigma\in\Sigma}, v_F)$ be an M2MA of dimension d. Let $\mathcal{V}$ denote the linear span of the reachable states of A, and $\mathcal{W}$ denote the linear span of the co-reachable states of A.

NBA Size	Mean	Median	Std Dev	Min	Max	Count
1	3.00	3.00	0.00	3	3	29
2	4.52	4.00	2.26	1	10	31
3	6.59	6.00	4.36	1	23	49
4	11.47	9.00	9.17	1	40	88
5	23.49	17.00	20.90	1	127	265
6	30.33	22.00	27.24	2	152	206
7	45.24	34.00	42.58	1	222	141
8	69.40	52.00	64.14	2	257	96
9	84.45	60.50	69.31	6	251	58
10	96.05	64.50	109.80	5	584	60

Fig. 11: NBA to learned M2MA

DBA Size	Mean	Median	Std Dev	Min	Max	Count
1	2.94	3.00	.24	2	3	17
2	5.29	5.00	2.07	2	9	24
3	9.30	8.00	5.73	2	29	57
4	15.71	13.50	12.04	1	65	118
5	27.90	21.00	23.85	1	121	250
6	36.69	24.00	37.99	1	212	209
7	49.96	33.00	50.38	1	257	179
8	69.12	49.00	67.33	1	279	101
9	102.02	91.00	74.48	2	312	81
10	116.78	121.00	72.01	4	337	37

Fig. 12: DBA to learned M2MA

n	Range of t
5	$[1, 5]$
10	$[2, 10]$
15	$[18, 22]$

(a) SUBA to minimized M2MA and DFA

n	Range of t
5	$[90, 140]$
6	$[160, 210]$
7	$[250, 350]$
8	$[350, 450]$
9	$[450, 550]$
10	$[580, 680]$

(b) NBA to learned M2MA

n	Range of t
5	$[0, 5]$
6	$[0, 6]$
7	$[0, 7]$
8	$[0, 8]$
9	$[0, 9]$
10	$[0, 10]$

(c) DBA to learned M2MA

Fig. 13: n and t experiment parameters

C.1 Computing a basis for the set of states

First we describe StateBasis, an algorithm to find a set of reachable states (and corresponding words reaching them) that form a basis for $\mathcal{V}$.

If the initial state v_I is the zero vector, then only the zero vector is reachable, and the resulting basis is the empty set. Otherwise, we initialize the sets B and C to contain just the pair (v_I, ε). Elements of B and C are pairs (v, x) where v is a reachable state of A and x is a word reaching v from the initial state.

While the set C is nonempty, we remove an element (v, x) from it. For each $\sigma \in \Sigma$, if $(v^\top \mu_\sigma)^\top$ is linearly independent of the set of vectors $\pi_1(B)$, then we add the pair $((v^\top \mu_\sigma)^\top, x\sigma)$ to both B and C. When C becomes empty, we return the set B. (See Algorithm 1.)

Because $\pi_1(B)$ is a linearly independent set of vectors of dimension d, the while loop terminates after adding at most d pairs to B. On termination, $\pi_1(B)$ is a basis for $\mathcal{V}$. The access words $\pi_2(B)$ form a prefix-closed set. Note that if the access words are processed in breadth-first order, a shortest accepted word (if any) is found.

Algorithm 1 Find a basis for the reachable states of A, with access words

```
function Algorithm StateBasis($A$)
    Assume $A = (\Sigma, v_I, \{\mu_\sigma\}_{\sigma \in \Sigma}, v_F)$
    if $v_I$ is the zero vector then return $\emptyset$, the empty basis
    Initialize $B := C := \{(v_I, \varepsilon)\}$
    while $C \neq \emptyset$ do
        $(v, x)$ := choose and remove an element of $C$
        for all $\sigma \in \Sigma$ do
            if $(v^\top \mu_\sigma)^\top$ is linearly independent of $\pi_1(B)$ then
                add the element $((v^\top \mu_\sigma)^\top, x\sigma)$ to $B$ and to $C$
    return $B$
```

To find a set $\tilde{B}$ of pairs (w, y) such that $w = \mu_y v_F$ and $\pi_1(\tilde{B})$ is a basis for $\mathcal{W}$, the linear span of the co-states of A, we define an analogous procedure, CoStateBasis, which on input A runs StateBasis on A^r, the reverse of A, and then reverses each access word.

C.2 Constructing a minimized M2MA

The algorithm MinimizeM2MA uses the information in B and $\tilde{B}$ (the bases and access words for the reachable states and co-reachable co-states of A) to construct an M2MA A' equivalent to A with the minimum possible dimension. Let B contain the pairs (v_i, x_i) for $i \in [1..d_1]$ and let $\tilde{B}$ contain the pairs (w_j, y_j) for $j \in [1..d_2]$, where $x_1 = y_1 = \varepsilon$. We define the $d \times d_1$ matrix V to consist of the columns v_i for $i \in [1..d_1]$ and the $d \times d_2$ matrix W to consist of the columns w_j for $j \in [1..d_2]$.

Consider the $d_1 \times d_2$ observation table T for $L(A)$ whose rows are indexed by x_i for $i \in [1..d_1]$ and whose columns are indexed by y_j for $j \in [1..d_2]$. Note that $T_{i,j} = 1$ iff $x_i \cdot y_j \in L(A)$ iff $v_I^\top \mu_{x_i} \mu_{y_j} v_F = 1$ iff $v_i^\top w_j = 1$, and thus $T = V^\top W$.

Let d_0 be the rank of the observation table T. By Lemma 3, any M2MA accepting $L(A)$ must have dimension at least d_0. We now construct an M2MA A' of dimension d_0 accepting $L(A)$.

Let i_k for $k \in [1..d_0]$ be the indices of a row basis of T, and let j_k for $k \in [1..d_0]$ be the indices of a column basis for T. Define the $d \times d_0$ matrix S to consist of the columns v_{i_k} for $k \in [1..d_0]$, and the $d \times d_0$ matrix E to consist of the columns w_{j_k} for $k \in [1..d_0]$. Thus the rows of $S^\top W$ are a row basis for T and the columns of $V^\top E$ are a column basis for T. If $T' = S^\top E$ then T' is a $d_0 \times d_0$ nonsingular submatrix of T corresponding to the rows i_k for $k \in [1..d_0]$ and the columns j_k for $k \in [1..d_0]$. We define $A' = (v'_I, \{\mu'_\sigma\}_{\sigma \in \Sigma}, v'_F)$, where $v'_I = (v_I^\top E)^\top$, v'_F has a 1 as its first entry and 0's elsewhere, and $\mu'_\sigma = (T')^{-1}(S^\top \mu_\sigma E)$. (See Algorithm 2.)

A key observation is that the columns of E are experiments that exactly characterize the languages accepted from states of A, that is, for any $v, v' \in \mathcal{V}$, $L_v(A) = L_{v'}(A)$ iff $v^\top E = (v')^\top E$.

Algorithm 2 Minimize the M2MA A

```
1  function ALGORITHM MINIMIZEM2MA($A$)
2      Assume $A = (\Sigma, v_I, \{\mu_\sigma\}_{\sigma \in \Sigma}, v_F)$
3      $B$ := STATEBASIS($A$)
4      $\tilde{B}$ := COSTATEBASIS($A$)
5      Assume $B = \{(v_i, x_i) \mid i \in [1..d_1]\}$, with $v_1 = \varepsilon$
6      Assume $\tilde{B} = \{(w_j, y_j) \mid j \in [1..d_2]\}$, with $y_1 = \varepsilon$
7      V := $d \times d_1$ matrix of columns $v_i$ for $i \in [1..d_1]$
8      W := $d \times d_2$ matrix of columns $w_j$ for $j \in [1..d_2]$
9      T := $V^\top W$, a $d_1 \times d_2$ matrix; let $d_0$ denote its rank
10     Find a subset of the rows of $T$ that are a row basis for $T$, and let their indices
           be $i_k$ for $k \in [1..d_0]$, with $i_1 = 1$
11     Find a subset of the columns of $T$ that are a column basis for $T$ and let their
           indices be $j_k$ for $k \in [1..d_0]$, with $j_1 = 1$
12     $S$ := $d \times d_0$ matrix of columns $v_{i_k}$ for $k \in [1..d_0]$
13     $E$ := $d \times d_0$ matrix of columns $w_{j_k}$ for $k \in [1..d_0]$
14     $T'$ := $S^\top E$, a $d_0 \times d_0$ (invertible) matrix
15     $v'_I$ := $v_I^\top E$
16     $v'_F$ := the dimension $d_0$ vector of one 1 followed by 0's
17     for all $\sigma \in \Sigma$ do
18         $\mu'_\sigma$ := $(T')^{-1}(S^\top \mu_\sigma E)$
19     $A'$ := $(\Sigma, v'_I, \{\mu'_\sigma\}_{\sigma \in \Sigma}, v'_F)$
20     return $A'$
```