AUTHORS	Pierre Marcus, Ilkka Törmä
TITLE	Descriptional Complexity of Winning Sets of Regular Languages
YEAR	2020
DOI	10.1007/978-3-030-62536-8_11
VERSION	Final draft
CITATION	Marcus P., Törmä I. (2020) Descriptional Complexity of Winning Sets of Regular Languages. In: Jirásková G., Pighizzini G. (eds) Descriptional Complexity of Formal Systems. DCFS 2020. Lecture Notes in Computer Science, vol 12442. Springer, Cham. https://doi.org/10.1007/978-3-030-62536-8_11

Descriptional Complexity of Winning Sets of Regular Languages

Pierre Marcus[1] and Ilkka Törmä[2][0000−0001−5541−8517]⋆

[1] M2 informatique fondamentale, École Normale Supérieure de Lyon, Lyon, France, `pierre.marcus@ens-lyon.fr`
[2] Department of Mathematics and Statistics, University of Turku, Turku, Finland, `iatorm@utu.fi` (corresponding author)

Abstract. We investigate certain word-construction games with variable turn orders. In these games, Alice and Bob take turns on choosing consecutive letters of a word of fixed length, with Alice winning if the result lies in a predetermined target language. The turn orders that result in a win for Alice form a binary language that is regular whenever the target language is, and we prove some upper and lower bounds for its state complexity based on that of the target language.

Keywords: State complexity · Regular languages · Winning sets

1 Introduction

Let us define a word-construction game of two players, Alice and Bob, as follows. Choose a set of binary words $L \subseteq \{0,1\}^*$ called the *target set*, a length $n \geq 0$ and a word $w \in \{A,B\}^n$ called the *turn order*, where A stands for Alice and B for Bob. The players construct a word $v \in \{0,1\}^n$ so that, for each $i = 0, 1, \ldots, n-1$ in this order, the player specified by w_i chooses the symbol v_i. If $v \in L$, then Alice wins the game, and otherwise Bob wins. The existence of a winning strategy for Alice depends on both the target set and the turn order. We fix the target set L and define its *winning set* $W(L)$ as the set of those words over $\{A,B\}$ that result in Alice having a winning strategy.

Winning sets were defined under this name in [9] in the context of symbolic dynamics, but they have been studied before that under the name of *order-shattering sets* in [1, 4]. The winning set has several interesting properties: it is downward closed in the index-wise partial order induced by $A < B$ (as changing B to A always makes the game easier for Alice) and it preserves the number of words of each length. This latter property was used in [8] to study the growth rates of substitutive subshifts.

If the language L is regular, then so is $W(L)$, as it can be recognized by an alternating finite automaton (AFA) [9], which only recognizes regular languages [3]. Thus we can view W as an operation on the class of binary regular languages, and in this article we study its state complexity in the general case

⋆ Author supported by Academy of Finland grant 295095.

and in several subclasses. In our construction the AFA has the same state set as the original DFA, so our setting resembles parity games, where two players construct a path in a finite automaton [10]. The main difference is that in a parity game, the player who chooses the next move is the owner of the current state, whereas here it is determined by the turn order word.

In the general case, the size of the minimal DFA for $W(L)$ can be doubly exponential in that of L. We derive a lower, but still superexponential, upper bound for bounded regular languages (languages that satisfy $L \subseteq w_1^* w_2^* \cdots w_k^*$ for some words w_i). We also study certain bounded permutation invariant languages, where membership is defined only by the number of occurrences of each symbol. In particular, we explicitly determine the winning sets of the languages $L_k = (0^*1)^k 0^*$ of words with exactly k occurrences of 1.

In this article we only consider the binary alphabet, but we note that the definition of the winning set can be extended to languages $L \subseteq \Sigma^*$ over an arbitrary finite alphabet Σ in a way that preserves the properties of downward closedness and $|L| = |W(L)|$. The turn order word is replaced by a word $w \in \{1, \ldots, |\Sigma|\}^*$. On turn i, Alice chooses a subset of size w_i of Σ, and Bob chooses the letter v_i from this set.

2 Definitions

We present the standard definitions and notations used in this article. For a set Σ, we denote by Σ^* the set of finite words over it, and the length of a word $w \in \Sigma^n$ is $|w| = n$. The notation $|w|_a$ means the number of occurrences of symbol $a \in \Sigma$ in w. The empty word is denoted by λ. For a language $L \subseteq \Sigma^*$ and $w \in \Sigma^*$, denote $w^{-1}L = \{v \in \Sigma^* \mid wv \in L\}$. We say L is *(word-)bounded* if $L \subseteq w_1^* \cdots w_k^*$ for some words $w_1, \ldots, w_k \in \Sigma^*$. Bounded languages have been studied from the state complexity point of view in [5].

A finite state automaton is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ where Q is a finite state set, Σ a finite alphabet, $q_0 \in Q$ the initial state, δ is the transition function and $F \subseteq Q$ is the set of final states. The language accepted from state $q \in Q$ is denoted $\mathcal{L}_q(\mathcal{A}) \subseteq \Sigma^*$, and the language of $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \mathcal{L}_{q_0}(\mathcal{A})$. The type of δ and the definition of $\mathcal{L}(\mathcal{A})$ depend on which kind of automaton $\mathcal{A}$ is.

– If $\mathcal{A}$ is a deterministic finite automaton, or DFA, then $\delta : Q \times \Sigma \to Q$ gives the next state from the current state and an input symbol. We extend it to $Q \times \Sigma^*$ by $\delta(q, \lambda) = q$ and $\delta(q, sw) = \delta(\delta(q, s), w)$ for $q \in Q$, $s \in \Sigma$ and $w \in \Sigma^*$. The language is defined by $\mathcal{L}_q(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q, w) \in F\}$.
– If $\mathcal{A}$ is a nondeterministic finite automaton, or NFA, then $\delta : Q \times \Sigma \to 2^Q$ gives the set of possible next states. We extend it to $Q \times \Sigma^*$ by $\delta(q, \lambda) = \{q\}$ and $\delta(q, sw) = \bigcup_{p \in \delta(q,s)} \delta(p, w)$ for $q \in Q$, $s \in \Sigma$ and $w \in \Sigma^*$. The language is defined by $\mathcal{L}_q(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$.
– If $\mathcal{A}$ is an alternating finite automaton, or AFA, then $\delta : Q \times \Sigma \to 2^{2^Q}$. We extend δ to $Q \times \Sigma^*$ by $\delta(q, \lambda) = \{S \subseteq Q \mid q \in S\}$ and $\delta(q, sw) = \{S \subseteq Q \mid \{p \in Q \mid S \in \delta(q, w)\} \in \delta(q, s)\}$ for $q \in Q$, $s \in \Sigma$ and $w \in \Sigma^*$. The language is defined by $\mathcal{L}_q(\mathcal{A}) = \{w \in \Sigma^* \mid F \in \delta(q, w)\}$.

All three types of finite automata recognize exactly the regular languages. An AFA can be converted into an equivalent NFA, and an NFA into a DFA, by the standard subset constructions. A standard reference for DFAs and NFAs is [6].

Two states $p, q \in Q$ of $\mathcal{A}$ are equivalent, denoted $p \sim q$, if $\mathcal{L}_p(\mathcal{A}) = \mathcal{L}_q(\mathcal{A})$. Every regular language $L \subseteq \Sigma^*$ is accepted by a unique DFA with the minimal number of states, which are all nonequivalent, and every other DFA that accepts L has an equivalent pair of states. Two words $v, w \in \Sigma^*$ are congruent by L, denoted $v \equiv_L w$, if for all $u_1, u_2 \in \Sigma^*$ we have $u_1 v u_2 \in L$ iff $u_1 w u_2 \in L$. They are right-equivalent, denoted $v \sim_L w$, if for all $u \in \Sigma^*$ we have $vu \in L$ iff $wu \in L$. The set of equivalence classes $\Sigma^*/{\equiv_L}$ is the syntactic monoid of L, and if L is regular, then it is finite. In that case the equivalence classes of $\sim_L$ can be taken as the states of the minimal DFA of L.

Let $\mathcal{P} : 2^{\Sigma^*} \to 2^{\Sigma^*}$ be a (possibly partially defined) operation on languages. The (regular) state complexity of $\mathcal{P}$ is $f : \mathbb{N} \to \mathbb{N}$, where $f(n)$ is the maximal number of states in a minimal automaton of $\mathcal{P}(\mathcal{L}(\mathcal{A}))$ for an n-state DFA $\mathcal{A}$.

We say that a function $f : \mathbb{N} \to \mathbb{R}$ grows *doubly exponentially* if there exist $a, b, c, d > 1$ with $a^{b^n} \leq f(n) \leq c^{d^n}$ for large enough n, and *superexponentially* if for all $a > 1$, $f(n) > a^n$ holds for large enough n.

3 Winning Sets

In this section we define winning sets of binary languages, present the construction of the winning set of a regular language, and prove some general lemmas. We defined the winning set informally at the beginning of Section 1. Now we give a more formal definition which does not explicitly mention games.

Definition 1 (Winning Set). *Let $n \in \mathbb{N}$ and $T \subseteq \{0,1\}^n$ be arbitrary. The* winning set *of T, denoted $W(T) \subseteq \{A,B\}^n$, is defined inductively as follows. If $n = 0$, then T is either the empty set or $\{\lambda\}$, and $W(T) = T$. If $n \geq 1$, then $W(T) = \{Aw \mid w \in W(0^{-1}T) \cup W(1^{-1}T)\} \cup \{Bw \mid w \in W(0^{-1}T) \cap W(1^{-1}T)\}$.*

For a language $L \subseteq \{0,1\}^*$, we define $W(L) = \bigcup_{n \in \mathbb{N}} W(L \cap \{0,1\}^n)$.

For Alice to win on a turn order of the form Aw, she has to choose either 0 or 1 as the first letter v_0 of the constructed word v, and then follow a winning strategy on the target set $v_0^{-1}T$ and turn order w. On a word Bw, Alice must have a winning strategy on $v_0^{-1}T$ and w no matter how Bob chooses v_0.

A language L over a linearly ordered alphabet Σ is *downward closed* if $v \in L$, $w \in \Sigma^{|v|}$ and $w_i \leq v_i$ for each $i = 0, \ldots, |v| - 1$ always implies $w \in L$.

Proposition 1 (Propositions 3.8 and 5.4 in [9]). *The winning set $W(L)$ of any $L \subseteq \{0,1\}^*$ is downward closed (with the ordering $A < B$) and satisfies $|W(L) \cap \{A,B\}^n| = |L \cap \{0,1\}^n|$ for all n. If L is regular, then so is $W(L)$.*

From a DFA $\mathcal{A}$, we can easily construct an alternating automaton for $W(\mathcal{A})$.

Definition 2 (Winning Set Automaton). *Let $\mathcal{A} = (Q, \{0,1\}, q_0, \delta, F)$ be a binary DFA. Define a "canonical" AFA for $W(\mathcal{L}(\mathcal{A}))$ as $(Q, \{A,B\}, q_0, \delta', F)$*

where $\delta'(q, A) = \{S \subset Q \mid \delta(q, 0) \in S \text{ or } \delta(q, 1) \in S\}$ *and* $\delta'(q, B) = \{S \subset Q \mid \delta(q, 0) \in S \text{ and } \delta(q, 1) \in S\}$*. This AFA clearly recognizes* $W(\mathcal{L}(\mathcal{A}))$*. We transform it into the equivalent NFA* $(2^Q, \{A, B\}, \{q_0\}, \delta'', 2^F)$*, where*

$$\delta''(S, A) = \{\{\delta(q, f(q)) \mid q \in S\} \mid f : S \to \{0, 1\}\}$$
$$\delta''(S, B) = \{\{\delta(q, b) \mid q \in S, b \in \{0, 1\}\}\}.$$

We usually work on the determinization of this NFA, which we denote by $W(\mathcal{A}) = (2^{2^Q}, \{A, B\}, \{\{q_0\}\}, \delta_W, F_W)$*. Here* $F_W = \{\mathsf{G} \in 2^{2^Q} \mid \exists S \in \mathsf{G} : S \subseteq F\}$ *and* $\delta_W(\mathsf{G}, c) = \bigcup_{S \in \mathsf{G}} \delta''(S, c)$ *for* $\mathsf{G} \subset 2^Q$ *and* $c \in \{A, B\}$*.*

Intuitively, as Alice and Bob construct a word, they also play a game on the states of $\mathcal{A}$ by choosing transitions. A state G of $W(\mathcal{A})$ is called a *game state*, and it represents a situation where Alice can force the game to be in one of the sets $S \in \mathsf{G}$, and Bob can choose the actual state $q \in S$. From the definition of the AFA it follows that exchanging the labels 0 and 1 on the two outgoing transitions of any one state of $\mathcal{A}$ does not affect δ'. In other words, the winning set of a DFA's language is independent of the labels of its transitions.

For example, take $L = 0^*1(0^*10^*1)0^*$, the language of words with an odd number of 1-symbols. Its winning set is $W(L) = (A + B)^*A$, as the last player has full control of the parity of occurrences of 1s. Figure 1 shows the minimal DFA for L and the NFA derived from it that recognizes $W(L)$. One can check that the language recognized by this NFA is indeed $(A+B)^*A$. Note how reading A lets each state of a set evolve independently by 0 or 1, while B makes both choices for all states simultaneously and results in one large set.

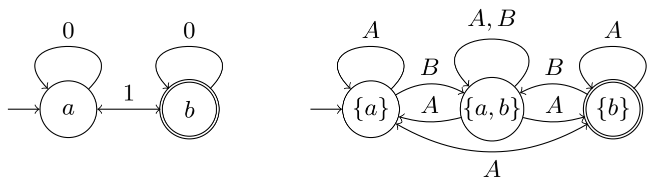

Fig. 1: A DFA for $L = 0^*1(0^*10^*1)0^*$, and the derived NFA for $W(L)$.

The following properties follow easily from the definition of $W(\mathcal{A})$.

Lemma 1. *Let* $\mathcal{A}$ *be a binary DFA,* $W(\mathcal{A})$ *the winning set DFA from Definition 2, and* δ_W *the iterated transition function for* $W(\mathcal{A})$*. Let* G *and* H *be game states of* $W(\mathcal{A})$*,* $R, S, T, V \subseteq 2^Q$ *sets of states, and* w *a word over* $\{A, B\}$*.*

(a) *Sets in game states evolve independently:* $\delta_W(\mathsf{G} \cup \mathsf{H}, w) = \delta_W(\mathsf{G}, w) \cup \delta_W(\mathsf{H}, w)$*.*
(b) *States in sets evolve almost independently: If* $S, R \subseteq Q$ *are disjoint, then* $\delta_W(\{S \cup R\}, w) = \{T \cup V \mid T \in \delta_W(\{S\}, w), V \in \delta_W(\{R\}, w)\}$*.*

(c) *Supersets can be removed from game states (since δ_W is monotone in its first argument by (a) and (b)): If $S, R \in \mathsf{G}$ and $S \subsetneq R$, then $\mathsf{G} \sim \mathsf{G} \setminus \{R\}$.*
(d) *Sets containing nonaccepting sink states can be removed from game states: If $S \in \mathsf{G}$ and some $q \in S$ has no path to a final state, then $\mathsf{G} \sim \mathsf{G} \setminus \{S\}$.*
(e) *Accepting sink states can be removed from sets: If $S \in \mathsf{G}$ and there is a sink state $q \in S \cap F$, then $\mathsf{G} \sim (\mathsf{G} \setminus \{S\}) \cup \{S \setminus \{q\}\}$.*

The next lemma helps prove equivalences of game states and words. The first part follows from monotonicity of δ_W, the second from Lemma 1(a).

Lemma 2. *Recall the assumptions of Lemma 1.*

(a) *Suppose that for each $S \in \mathsf{G}$ there is $R \in \mathsf{H}$ with $R \subseteq S$, and reciprocally. Then $\mathsf{G} \sim \mathsf{H}$.*
(b) *Let $v, w \in \{A, B\}^*$. If for all $q \in Q$, the game states $\delta_W(\{\{q\}\}, v)$ and $\delta_W(\{\{q\}\}, w)$ are either both accepting or both rejecting, then $v \equiv_{W(\mathcal{L}(\mathcal{A}))} w$.*

Recall the *Dedekind numbers* $D(n)$, which count the number of antichains of subsets of $\{1, \ldots, n\}$ with respect to set inclusion. Their growth is doubly exponential: $a^{a^n} < D(n) < b^{b^n}$ holds for large enough n if $a < 2 < b$. This follows from $\binom{n}{\lceil n/2 \rceil} \leq \log_2 D(n) \leq (1 + O(\log n/n))\binom{n}{\lceil n/2 \rceil}$ [7] and the well known asymptotic formula $\binom{n}{\lceil n/2 \rceil} = \Theta(2^n/\sqrt{n})$.

Proposition 2. *Let $\mathcal{A}$ an n-state DFA. The number of states in the minimal DFA for $W(\mathcal{L}(\mathcal{A}))$ is at most the Dedekind number $D(n)$.*

Proof. Every game state is equivalent to an antichain by Lemma 1(c), so the number of nonequivalent game states is at most $D(n)$. □

We have computed the exact state complexity of the winning set operation for DFAs with at most 5 states; the 6-state case is no longer feasible with our program and computational resources. The sequence begins with $1, 4, 16, 62, 517$.

4 Doubly Exponential Lower Bound

In this section we construct a family of automata for which the number of states in the minimal winning set automaton is doubly exponential. The idea is to reach any desired antichain of subsets of a special subset of states, and then to make sure these game states are nonequivalent. To do this we split the automaton into several components. First we present a "subset factory gadget" that allows to reach any set of the form $\{S\}$ where S is a subset of a specific length-n path in the automaton. This gadget will be used several times to accumulate subsets in the game state. Then we present a "testing gadget" that lets us distinguish between game states by whether they contain a (subset of a) given set or not.

Recall that the transition labels of a binary DFA are irrelevant to the winning set of its language. In this section we define automata by describing their graphs, and a node with two outgoing transitions can have them arbitrary labeled by 0 and 1. Incoming and outgoing transitions in the figures indicate how the gadgets connect to the rest of the automaton.

Lemma 3 (Subset factory gadget). *Let* $\mathrm{GenSubset}_n$ *be the graph in Figure 2. For* $i \in \{1, \dots, n\}$*, denote* $o_i = e_{2n+i-1}$ *(the* n *rightmost states labeled by* e*). For* $S \subseteq \{1, \dots, n\}$*, let* w_S^{gen} *be the concatenation* $w_1 w_2 \dots w_n$ *where* $w_i = BA$ *if* $i \in S$*, and* $w_i = AB$ *if* $i \notin S$*. Then* $\delta_W(\{\{b_1\}\}, w_S^{\mathrm{gen}})) \sim \{\{o_i \mid i \in S\}\}$ *for each binary DFA that contains* $\mathrm{GenSubset}_n$ *as a subgraph.*

The idea is that at step i, reading B adds c_i to each subset of the game state, and then reading A avoids the sink s_i. On the other hand reading A creates two versions of each subset of the game state, one that continues on the upper row, and one that falls into the sink s_i when B is read, and can then be ignored.

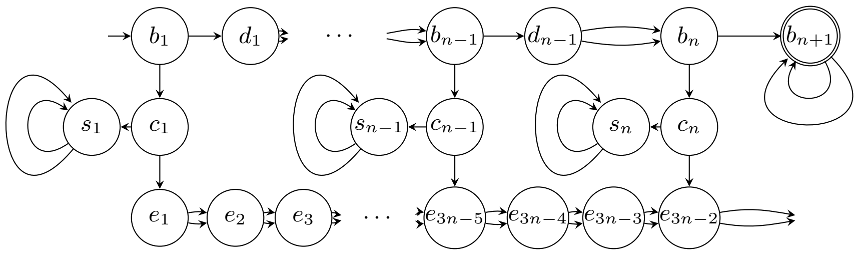

Fig. 2: $\mathrm{GenSubset}_n$, the subset factory gadget.

Lemma 4 (Game state factory gadget). *Let* $\mathrm{GenState}_n$ *be the graph in Figure 3 and* $\mathcal{A}$ *any DFA over* $\{0,1\}$ *that contains it. For all* $\mathsf{G} = \{S_1, \dots, S_\ell\}$ *where each* $S_i \subseteq \{r_1, \dots, r_n\}$*, let* $w_{\mathsf{G}}^{\mathrm{gen}} \in \{A, B\}^{\ell(3n+1)}$ *be the concatenation of* $Aw_{S_i}^{\mathrm{gen}} A^n$ *for* $i \in \{0, \dots, \ell\}$*. Then* $\delta_W(\{\{a_1\}\}, w_{\mathsf{G}}^{\mathrm{gen}}) \sim \mathsf{G} \cup \{\{a_1\}\} \cup \mathsf{G}'$ *for some game state* G' *that does not contain a subset of the states of* $\mathrm{GenState}_n$*.*

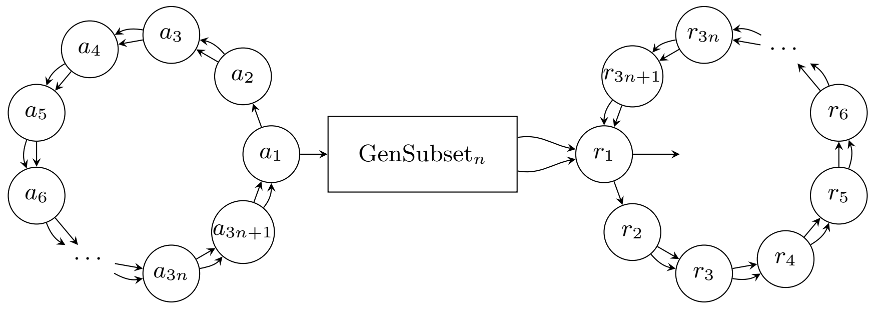

Fig. 3: $\mathrm{GenState}_n$, the game state factory gadget.

The idea is to successively add new sets S_i to the game state, while previously made subsets will wait by rotating in the r-cycle. A singleton set rotates in the a-cycle so that reading A from the state a_1 creates a new singleton set in the subset factory gadget. The word $w^{\text{gen}}_{S_i}$ transforms it into a set of the correct form, and then reading A^n both moves this new subset to the r-cycle with the previously created sets and rotates the singleton set back to a_i.

Lemma 5 (Testing gadget). *Let* Testing_n *be the graph in Figure 4.*

(a) *For* $P \subseteq \{1, \ldots, n\}$*, define* $w^{\text{test}}_P \in \{A, B\}^n$ *by* $w^{\text{test}}_P[i] = A$ *iff* $n - i + 1 \in P$*. Then for each* $I \subseteq \{1, \ldots, n\}$*, the game state* $\delta_W(\{\{q_i \mid i \in I\}\}, w^{\text{test}}_P)$ *is accepting iff* $I \subseteq P$*.*

(b) *Let* V *be the set of nodes of the graph* Testing_n*. Then for all* $\mathsf{G} \in 2^{2^V}$ *and* $w \in \{A, B\}^{\geq 2n}$*, the game state* $\delta_W(\mathsf{G}, w)$ *is not accepting.*

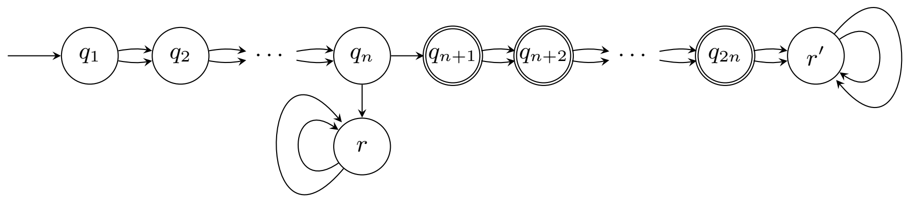

Fig. 4: Testing_n, the testing gadget.

The idea is that reading A or B moves the game state toward r', except when the set contains the state q_n and B is read, causing it to fall into the sink r.

Theorem 1. *For each* $n > 0$ *there exists a DFA* $\mathcal{A}_n$ *over* $\{0, 1\}$ *with* $15n + 3$ *states such that the minimal DFA for* $W(\mathcal{L}(\mathcal{A}_n))$ *has a least* $D(n)$ *states.*

Together with Proposition 2, this implies that the state complexity of W restricted to regular languages grows doubly exponentially.

Proof (sketch). Let $\mathcal{A}_n$ be the DFA obtained by combining Testing_n with the outgoing arrow of GenState_n and assigning a_1 as the initial state. For an antichain G on the powerset of $\{r_1, \ldots, r_n\}$, let $X_\mathsf{G} = \delta_W(\{\{a_1\}\}, w^{\text{gen}}_\mathsf{G})$. Lemma 4 gives $X_\mathsf{G} \sim \{\{a_1\}\} \cup \mathsf{G} \cup \mathsf{G}'$ where each set in G' contains a state of Testing_n.

We show that distinct antichains G result in nonequivalent states. Let $P \subseteq \{1, \ldots, n\}$ and consider $X'_\mathsf{G} = \delta_W(X_\mathsf{G}, AB^{2n}A^{2n+1}w^{\text{test}}_P)$. We claim that X'_G is accepting iff some element of G is a subset of $\{r_i \mid i \in P\}$. By Lemma 1(a) we may analyze the components of X_G separately.

- We have $\delta_W(\{\{a_1\}\}, A) = \{\{a_2\}, \{b_1\}\}$. The part $\{b_1\}$ is destroyed by the sink state s_1 when we read Bs, and the part $\{a_2\}$ rotates in the a-cycle without encountering accepting states.
- Each set of G' contains a state of Testing$_n$, which will reach one of the nonaccepting sinks r or r'.
- The game state $\delta_W(\mathsf{G}, AB^{2n}A^{2n+1})$ consists of the sets $\{q_i \mid r_i \in S\}$ for $S \in \mathsf{G}$, as well as sets that contain at least one element of $\{r_2, \ldots, r_{n+1}\}$. The latter will rotate in the r-cycle. By Lemma 5, the former sets produce an accepting game state in X'_{G} iff some $S \in \mathsf{G}$ is a subset of $\{r_i \mid i \in P\}$.

We have found $D(n)$ nonequivalent states in $W(\mathcal{A})$. □

5 Case of the Bounded Regular Languages

In this section we prove an upper bound on the complexity of the winning set of a bounded regular language. Our motivation comes from the fact that bounded regular languages correspond to so-called *zero entropy sofic shifts* in symbolic dynamics, which are defined by the number of words of given length that occur in them, and the fact that the winning set operation preserves this number. Our proof technique is based on tracing the evolution of individual states of a DFA $\mathcal{A}$ in the winning set automaton $W(\mathcal{A})$ when reading several A-symbols in a row.

Definition 3 (Histories of Game States). *Let $\mathcal{A} = (Q, \{0,1\}, q_0, \delta, F)$ be a DFA. Let $\mathsf{G} \in 2^{2^Q}$ be a game state of $W(\mathcal{A})$, and for each $i \geq 0$, let $\mathsf{G}_i \sim \delta_W(\mathsf{G}, A^i)$ be the game state with all supersets removed as per Lemma 1(c). A* history function *for G is a function h that associates to each $i > 0$ and each set $S \in \mathsf{G}_i$ a* parent set *$h(i, S) \in \mathsf{G}_{i-1}$, and to each state $q \in S$ a set of* parent states *$h(i, S, q) \subseteq h(i, S)$ such that*

- *$\{q\} \in \delta_W(\{h(i, S, q)\}, A)$ for all $q \in S$, and*
- *$h(i, S)$ is the disjoint union of $h(i, S, q)$ for $q \in S$.*

Note that this implies $S \in \delta_W(\{h(i, S)\}, A)$ for each i.

The history *of a set $S \in \mathsf{G}_i$ from i under h is the sequence $S_0, S_1, \ldots, S_i = S$ with $S_{j-1} = h(j, S_j)$ for all $0 < j \leq i$. A history of a state $q \in S$ in S under h is a sequence $q_0, \ldots, q_i = q$ with $q_{j-1} \in h(j, S_j, q_j)$ for all $0 < j \leq i$.*

Every game state has at least one history function: each $S \in \mathsf{G}_i$ has at least one set $R \in \mathsf{G}_{i-1}$ with $S \in \delta_W(\{R\}, A)$, so we can choose $R = h(i, S)$, and similarly for the $h(i, S, q)$. It can have several different history functions, and each of them defines a history for each set S. A state of S can have several histories under a single history function.

For the rest of this section, we fix an n-state DFA $\mathcal{A} = (Q, \{0,1\}, q_0, \delta, F)$ that recognizes a bounded binary language and has disjoint cycles. Let the lengths of the cycles be $k_1, \ldots, k_p$, and let ℓ be the number of states not part of any cycle.

We define a preorder $\leq$ on the state set Q by reachability: $p \leq q$ holds if and only if there is a path from p to q in $\mathcal{A}$. For two history functions h, h' of a game state G, we write $h \leq h'$ if for each $i > 0$, each $S \in \mathsf{G}_i$ and each $q \in S$, there exists a function $f : h(i, S, q) \to h'(i, S, q)$ with $p \leq f(p)$ for all $p \in h(i, S, q)$. This defines a preorder on the set of history functions of G. We write $h < h'$ if $h \leq h'$ and $h' \not\leq h$. A history function h is *minimal* if there exists no history function h' with $h' < h$. Intuitively, a minimal history function is one where the histories of states stay in the early cycles of $\mathcal{A}$ as long as possible. Since the choices of $h(i, S)$ and $h(i, S, q)$ can be made independently, minimal history functions always exist.

Lemma 6. *Let $\mathsf{G} \in 2^{2^Q}$ be any game state of $W(\mathcal{A})$. Then there exist $k \leq \mathrm{lcm}(k_1, \ldots, k_p) + 2n + \max_{x \neq y} \mathrm{lcm}(k_x, k_y)$ and $m \leq \mathrm{lcm}(k_1, \ldots, k_p)$ such that $\delta_W(\mathsf{G}, A^k) \sim \delta_W(\mathsf{G}, A^{k+m})$.*

The idea of the proof is that under a minimal history function, no state $q \in S \in \mathsf{G}$ can spend too long in a cycle it did not start in: the maximal number of steps is comparable to the lcm of the lengths of successive cycles.

Theorem 2. *Let $\mathcal{A}$ be an n-state binary DFA whose language is bounded. Then there is a partition $\ell + k_1 + \cdots + k_p = n$ such that the minimal DFA for $W(\mathcal{L}(\mathcal{A}))$ has at most $\sum_{m=0}^{\ell+p+1} (p \cdot \max_{x \neq y} \mathrm{lcm}(k_x, k_y) + 2\ell + 2\mathrm{lcm}(k_1, \ldots, k_p))^m$ states.*

Proof. Denote the minimal DFA for $W(\mathcal{L}(\mathcal{A}))$ by $\mathcal{B}$. We may assume that $\mathcal{A}$ is minimal, and then it has disjoint cycles, as otherwise the number of length-n words in $\mathcal{L}(\mathcal{A})$ would grow exponentially while in a bounded language this growth is at most polynomial. Let $k_1, \ldots, k_p$ be the lengths of the cycles and ℓ the number of remaining states, and denote $P = p \cdot \max_{x \neq y} \mathrm{lcm}(k_x, k_y) + 2\ell + 2\mathrm{lcm}(k_1, \ldots, k_p)$. Then any $w \in \mathcal{L}(W(\mathcal{A}))$ has $|w|_B \leq \ell + p$, as otherwise Bob can win by choosing to leave a cycle whenever possible.

Consider a word $w = A^{t_0} B A^{t_1} B \cdots B A^{t_m}$ with $0 \leq m \leq \ell + p$. If $t_i \geq P$ for some i, then Lemma 6 implies $\delta_W(\mathsf{G}, A^{t_i}) \sim \delta_W(\mathsf{G}, A^t)$ for the game state $\mathsf{G} = \delta_W(\{\{q_0\}\}, A^{t_0} B \cdots A^{t_{i-1}} B)$ and some $t < t_i$. Thus the number of distinct states of $\mathcal{B}$ reachable by such words is at most P^{m+1}. The claim follows. ☐

The upper bound we obtain (the maximum of the expression taken over all partitions of n) is at least n^n. We do not know whether the actual complexity is superexponential for bounded languages. If we combine the gadgets $\mathrm{GenSubset}_n$ and $\mathrm{Testing}_n$, the resulting DFA recognizes a language whose winning set requires at least 2^n states, so for finite (and thus bounded) regular languages the state complexity of the winning set is at least exponential.

6 Chain-Like Automata

In this section we investigate a family of binary automata consisting of a chain of states with a self-loop on each state. More formally, define a 1-*bounded chain DFA* as $\mathcal{A} = (Q, \{0,1\}, q_0, \delta, F)$ where $Q = \{0, 1, \ldots, n-1\}$, $q_0 = 0$, $\delta(i, 0) = i$ and $\delta(i, 1) = i + 1$ for all $i \in Q$ except $\delta(n-1, 1) = n - 1$, and $n - 1 \notin F$. See

Figure 5 for an example. It is easy to see that these automata recognize exactly the regular languages L such that $w \in L$ depends only on $|w|_1$, and $|w|_1$ is also bounded. Of course, the labels of the transitions have no effect on the winning set $W(\mathcal{L}(\mathcal{A}))$ so the results of this section apply to every DFA with the structure of a 1-bounded chain DFA.

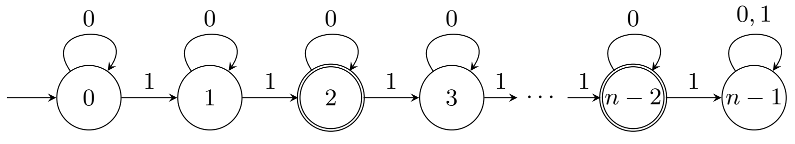

Fig. 5: A 1-bounded chain DFA. Any states except $n-1$ can be final.

Lemma 7. *Let $\mathcal{A}$ be an n-state 1-bounded chain DFA. Let $\equiv$ stand for $\equiv_{W(\mathcal{L}(\mathcal{A}))}$.*

(a) *For every state $q \in Q$ and every $S \in \delta_W(\{\{q\}\}, AB)$, there exists $R \in \delta_W(\{\{q\}\}, BA)$ with $R \subseteq S$.*
(b) *For all $k \in \mathbb{N}$, $B^k A^k B^{k+1} \equiv B^{k+1} A^k B^k$.*
(c) *For all $k \in \mathbb{N}$, $A^{k+1} B^k A^k \equiv A^k B^k A^{k+1}$.*
(d) *$A^{n-1} \equiv A^n$ and $B^{n-1} \equiv B^n$.*

The intuition for (a) is that the turn order BA is better for Alice than AB, since she can undo any damage Bob just caused. The other items are proved by concretely analyzing the evolution of game states, which A intuitively "moves around with precise control" and B "thickens". Lemma 2 simplifies the analysis.

Theorem 3. *Let $\mathcal{A}$ be a 1-bounded chain DFA with n states. The number of states in the minimal DFA of $W(\mathcal{L}(\mathcal{A}))$ is $O(n^{1/5} e^{4\pi\sqrt{\frac{n}{3}}})$.*

Proof. Since $\mathcal{A}$ does not accept any word with n or more 1-symbols, $W(\mathcal{L}(\mathcal{A}))$ contains no word with n or more B-symbols. Lemma 7(b) and (c) allow us to rewrite every word of $W(\mathcal{L}(\mathcal{A}))$ in the form $A^{n_1} B^{n_2} A^{n_3} B^{n_4} \cdots A^{n_{2r-1}} B^{n_{2r}}$ where the sequence $n_1, \ldots, n_{2r}$ is first nondecreasing and then nonincreasing, and $n_2 + n_4 + \cdots + n_{2r} < n$. With Lemma 7(d) we can also guarantee $n_1, n_3, \ldots, n_{2r-1} < n$, so that $\sum_i n_i < 4n$. In [2], Auluck showed that the number $Q(m)$ of partitions $m = n_1 + \ldots n_r$ of an integer m that are first nondecreasing and then nonincreasing is $\Theta(m^{-4/5} e^{2\pi\sqrt{m/3}})$. Of course, $v \equiv w$ implies $v \sim w$. Thus the number of non-right-equivalent words for $W(\mathcal{L}(\mathcal{A}))$, and the number of states in its minimal DFA, is at most $1 + \sum_{m=0}^{4n-1} Q(m) = O(n^{1/5} e^{4\pi\sqrt{\frac{n}{3}}})$. □

7 Case Study: Exact Number of 1-Symbols

In the previous section we bounded the complexity of the winning set of certain bounded permutation invariant languages. Here we study a particular case, the language of words with exactly n ones, or $L = (0^*1)^n0^*$. We not only compute the number of states in the minimal automaton (which is cubic in n), but also describe the winning set. Throughout the section $\mathcal{A}$ is the minimal automaton for L, described in Figure 6. For $S \subseteq Q$, we denote $\overline{S} = \{\min(S), \min(S)+1, \dots, \max(S)\}$, and for any game state G of $W(\mathcal{A})$, denote $\overline{\mathsf{G}} = \{\overline{S} \mid S \in \mathsf{G}\}$.

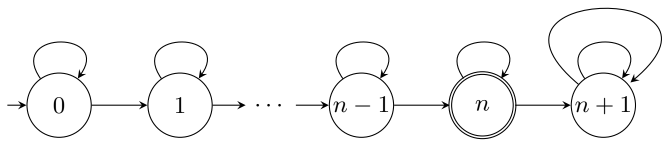

Fig. 6: The minimal DFA for $L = (0^*1)^n0^*$.

Lemma 8. *Each game state* G *of* $W(\mathcal{A})$ *is equivalent to* $\overline{\mathsf{G}}$.

The idea is that the left and right ends of sets in G evolve independently of their other elements, and their positions determine whether G is final.

Lemma 9. *Let* T *be the set of integer triples* (i, ℓ, N) *with* $0 \leq i \leq n$, $1 \leq \ell \leq n - i + 1$ *and* $1 \leq N \leq n - i - \ell + 2$. *For* $(i, \ell, N) \in T$, *let*

$$\mathsf{G}(i, \ell, N) := \{\{i, \dots, i+\ell-1\}, \{i+1, \dots, i+\ell\}, \dots, \{i+N-1, \dots, i+\ell+N-2\}\}.$$

(a) *Each reachable game state of* $W(\mathcal{A})$ *is equivalent to some* $\mathsf{G}(i, \ell, N)$ *for* $(i, \ell, N) \in T$, *or to* $\emptyset$.
(b) *The game states* $\mathsf{G}(i, \ell, N)$ *for* $(i, \ell, N) \in T$ *are nonequivalent.*
(c) *Every* $\mathsf{G}(i, \ell, N)$ *for* $(i, \ell, N) \in T$ *is equivalent to some reachable game state.*

The game state $\mathsf{G}(i, \ell, N)$ is an interval of intervals, where i is the leftmost position of the first interval, ℓ is their common length, and N is their number. The first item is proved by induction, and the others by exhibiting a word over $\{A, B\}$ that produces or separates given game states. Then the minimal DFA for $W(L)$ has $|T| + 1$ states, and counting them yields the following.

Proposition 3. *The minimal DFA for* $W(L)$ *has* $\frac{n^3}{6} + n^2 + \frac{11n}{6} + 2$ *states.*

Proposition 4. $W(L)$ *is exactly the set of words* $w \in \{A, B\}^*$ *such that* $|w|_A \geq n$, $|w|_B \leq n$, *and every suffix* v *of* w *satisfies* $|v|_A \geq |v|_B$.

Proof (sketch). Every $w \in W(L)$ satisfies $|w|_A \geq n$ since Bob can play only 0s, and $|w|_B \leq 1$ since he can play only 1s. Only game states of the form $\mathsf{G}(i, 1, N)$ can be accepting. Since reading A decreases the parameter ℓ by one, and B increases ℓ by one, words of $W(L)$ must have after each B an associated A somewhere in the word. This is equivalent to the suffix condition. Conversely, on words of the given form Alice can win by associating to each $w_i = B$ some $w_j = A$ that occurs after it, choosing $v_j \neq v_i$ for the constructed word $v \in \{0, 1\}^*$, and choosing the remaining symbols so that $|v|_1 = n$. □

8 A Context-Free Language

In this section we prove that the winning set operator does not in general preserve context-free languages by studying the winning set of the Dyck language $D \subseteq \{0,1\}^*$ of balanced parentheses. In our formalism, 0 stands for an opening parenthesis and 1 for a closing parenthesis.

Proposition 5. *The winning set of the Dyck language is not context-free.*

Proof (sketch). Take $L = W(D) \cap (AA)^*(BB)^*(AA)^*$, which is context-free if $W(D)$ is. We claim that $L = \{A^{2i}B^{2j}A^{2k} \mid i \geq j, k \geq 2j\}$. First, if Bob closes $2j$ parentheses, Alice must open at least $2j$ parentheses beforehand, so $i \geq j$ is necessary. If Bob opens $2j$ parentheses instead, when Alice plays a second time, she has to close $4j$ parentheses, hence $k \geq 2j$. Thus the right hand side contains L. Conversely, Alice can win on $A^{2i}B^{2j}A^{2k}$ by leaving exactly $2j$ parentheses open before Bob's turns and then closing all open parentheses, so L contains the right hand side. It's a standard exercise to prove that L is not context-free. □

References

1. Anstee, R., Rónyai, L., Sali, A.: Shattering News. Graphs and Combinatorics **18**(1), 59–73 (2002). https://doi.org/10.1007/s003730200003
2. Auluck, F.: On some new types of partitions associated with generalized Ferrers graphs. In: Mathematical Proceedings of the Cambridge Philosophical Society. vol. 47, pp. 679–686. Cambridge University Press (1951). https://doi.org/10.1017/S0305004100027134
3. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. Journal of the ACM **28**(1), 114–133 (1981). https://doi.org/10.1145/322234.322243
4. Friedl, K., Rónyai, L.: Order shattering and Wilson's theorem. Discrete Mathematics **270**(1), 127–136 (2003). https://doi.org/10.1016/S0012-365X(02)00869-5
5. Herrmann, A., Kutrib, M., Malcher, A., Wendlandt, M.: Descriptional Complexity of Bounded Regular Languages. In: Câmpeanu, C., Manea, F., Shallit, J. (eds.) Descriptional Complexity of Formal Systems. pp. 138–152. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-41114-9_11
6. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation (3rd Edition). Addison-Wesley Longman Publishing Co., Inc., USA (2006)
7. Kleitman, D., Markowsky, G.: On Dedekind's Problem: The Number of Isotone Boolean Functions. II. Transactions of the American Mathematical Society **213**, 373–390 (1975). https://doi.org/10.2307/1998052
8. Peltomäki, J., Salo, V.: On winning shifts of marked uniform substitutions. RAIRO-Theoretical Informatics and Applications **53**(1-2), 51–66 (2019). https://doi.org/10.1051/ita/2018007
9. Salo, V., Törmä, I.: Playing with Subshifts. Fundamenta Informaticae **132**(1), 131–152 (2014). https://doi.org/10.3233/FI-2014-1037
10. Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science **200**(1), 135–183 (1998). https://doi.org/10.1016/S0304-3975(98)00009-7