Winning Sets of Regular Languages:
Descriptive and Computational Complexity

Pierre Marcus
M2 informatique fondamentale
École Normale Supérieure de Lyon, Lyon, France
pierre.marcus@ens-lyon.fr

Ilkka Törmä∗
Department of Mathematics and Statistics
University of Turku, Turku, Finland
iatorm@utu.fi

May 4, 2021

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. We also consider the computational complexity of membership and intersection problems of winning sets.

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 [13] in the context of symbolic dynamics, but they have been studied before that under the name of order-shattering sets in [1, 8]. The winning set has several interesting properties: it is downward closed in the index-wise partial order induced by $A < B$ (as changing

∗Author supported by Academy of Finland grant 295095.
$B$ 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 \[12\] 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) \[13\], which only recognizes regular languages \[4\]. 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 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 \[14\]. 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 $L \subseteq w_1 \cdot w_2 \cdots w_k$ for some words $w_1$). 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 $L_k = (0^*1)^k0^*$ of words with exactly $k$ occurrences of $1$. In the final section we consider the computational complexity of determining membership in $W(L)$ and whether a given regular language intersects it, which are P-complete and PSPACE-complete respectively when regular languages are given as finite automata.

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 $\Sigma$ 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 $\Sigma$, and Bob chooses the letter $v_i$ from this set.

This article is an extended version of the conference publication \[10\].

## 2 Definitions

We present the standard definitions and notations used in this article. For a set $\Sigma$, we denote by $\Sigma^*$ the set of finite words over it, and the length of a word $w \in \Sigma^*$ 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 $\lambda$. For a language $L \subseteq \Sigma^*$ and $w \in \Sigma^*$, denote $w^{-1}L = \{v \in \Sigma^* \mid uv \in L\}$. We say $L$ is (word-)bounded if $L \subseteq w_1 \cdot w_2 \cdots w_k$ for some words $w_1, \ldots, w_k \in \Sigma^*$. Bounded languages have been studied from the state complexity point of view in \[6\].

A finite state automaton is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ where $Q$ is a finite state set, $\Sigma$ a finite alphabet, $q_0 \in Q$ the initial state, $\delta$ is the transition function and $F \subseteq Q$ is the set of final states. The language accepted from state $q \in Q$ is denoted $L_q(\mathcal{A}) \subseteq \Sigma^*$, and the language of $\mathcal{A}$ is $L(\mathcal{A}) = L_{q_0}(\mathcal{A})$. The type of $\delta$ and the definition of $L(\mathcal{A})$ 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 $L_q(\mathcal{A}) = \{ w \in \Sigma^* \mid \delta(q, w) \in F\}$. 

2
• 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 $L_p(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$. An NFA can be converted into an equivalent DFA by the standard subset constructions. A standard reference for DFAs and NFAs is [7].

Example 2. If $W$ is a language, we give a more formal definition which does not explicitly mention games. We defined the winning set informally at the beginning of Section 1. Now we give a formal definition of the winning set of a regular language, and prove some general lemmas.

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 $\{x\}$, 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 $v_0$.

Example 2. Consider the regular language $L = (0+1)^*011(0+1)^*$ of all binary words that contain an occurrence of 011, and the turn order $w = AABAAB$. We claim that Alice has a winning strategy on $w$, so that $w \in W(L)$. On her first two turns she can play 01, after which it is Bob’s turn. If Bob plays a 1, the constructed word $v \in \{0,1\}^6$ begins with 011. If Bob plays a 0 instead, Alice can follow with 11, so $v$ begins with 01011. In both cases $v \in L$ and Alice wins. Note how the final $B$ plays no role in this analysis, and indeed $AABAAB(A+B)^* \subseteq W(L)$.

A language $L$ over a linearly ordered alphabet $\Sigma$ 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 $w \in L$. 3
Proposition 3 (Propositions 3.8 and 5.4 in [13]). 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 a nondeterministic automaton for \( W(\mathcal{A}) \).

Definition 4 (Winning Set Automaton). Let \( \mathcal{A} = (Q, \{0,1\}, q_0, \delta, F) \) be a binary DFA. Define a “canonical” NFA for the winning set \( W(L(\mathcal{A})) \) as \( \mathcal{A}' = (2^Q, \{A,B\}, \{q_0\}, \delta', 2^F) \), where

\[
\delta'(S,A) = \{\delta(q,f(q)) | q \in S \} \quad | \quad f : S \to \{0,1\}
\]

\[
\delta'(S,B) = \{\delta(q,b) | q \in S, b \in \{0,1\}\}.
\]

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 \( F_W = \{G \in 2^{2^Q} | \exists S \subseteq G : S \subseteq F \} \) and \( \delta_W(G,c) = \bigcup_{S \subseteq G} \delta'(S,c) \) for \( G \subseteq 2^Q \) and \( c \in \{A,B\} \).

Lemma 5. In the situation of Definition 4, we have \( L(\mathcal{A}') = W(L(\mathcal{A})) \).

Proof. Let \( S \in 2^Q \) and \( w \in \{A,B\}^* \) be arbitrary. We prove by induction on \(|w|\) that \( w \in L(\mathcal{A}') \) if and only if \( w \in \bigcap_{q \in S} W(L_q(\mathcal{A})) \). The result follows by considering \( S = \{q_0\} \). The case base \( w = \lambda \) is clear, since the final states of \( \mathcal{A}' \) are exactly the subsets of \( F \).

Consider \( w = Aw \). If \( Aw \in L_S(\mathcal{A}') \) then there is a function \( f : S \to \{0,1\} \) with \( v \in L_T(\mathcal{A}') \) for \( T = \{\delta(q,f(q)) | q \in S\} \). By the induction hypothesis, for each \( q \in S \) we have \( v \in W(L_{\delta(q,f(q))}(\mathcal{A})) = W(f(q)^{-1}L_q(\mathcal{A})) \), so that \( Av \in W(L_q(\mathcal{A})) \). Conversely, if \( Av \in W(L_q(\mathcal{A})) \), then \( v \in W(0^{-1}L_q(\mathcal{A})) \cup W(1^{-1}L_q(\mathcal{A})) = W(L_{\delta(q,0)}(\mathcal{A})) \cup W(L_{\delta(q,1)}(\mathcal{A})) \), so we may choose \( f(q) \in \{0,1\} \) with \( v \in W(L_{\delta(q,f(q))}(\mathcal{A})) \). By the induction hypothesis, \( v \in L_T(\mathcal{A}') \), and then \( Av \in L_S(\mathcal{A}') \).

Consider then \( w = Bw \). We have \( Bw \in L_S(\mathcal{A}') \) if and only if \( v \in L_T(\mathcal{A}') \) with \( T = \{\delta(q,b) | q \in S, b \in \{0,1\}\} \). By the induction hypothesis this is equivalent to \( v \in \bigcap_{b \in \{0,1\}} W(L_{\delta(q,b)})(\mathcal{A})) = \bigcap_{b \in \{0,1\}} W(b^{-1}L_q(\mathcal{A})) \), hence to \( Bw \in W(L_q(\mathcal{A})) \), for each \( q \in S \).

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 \( S \in G \), and Bob can choose the actual state \( q \in S \). From the definition of \( \mathcal{A}' \) it follows that exchanging the labels 0 and 1 on the two outgoing transitions of any one state of \( \mathcal{A} \) does not affect \( \delta' \). In other words, the winning set of a DFA’s language is independent of the labels of its transitions.

Example 6. Consider \( L = 0^*1(0^*10^*1)^* \), 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 7 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.

The following properties follow easily from the definition of \( W(\mathcal{A}) \).
Lemma 7. Let $A$ be a binary DFA, $W(A)$ the winning set DFA from Definition 4, and $\delta_W$ the iterated transition function for $W(A)$. Let $G$ and $H$ be game states of $W(A)$, $R, S, T, V \subseteq 2^Q$ sets of states, and $w$ a word over $\{A, B\}$.

(a) Sets in game states evolve independently: $\delta_W(G \cup H, w) = \delta_W(G, w) \cup \delta_W(H, w)$.

(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: If $S, R \in G$ and $S \subseteq R$, then $G \sim G \setminus \{R\}$.

(d) Sets containing nonaccepting sink states can be removed from game states: If $S \in G$ and some $q \in S$ has no path to a final state, then $G \sim G \setminus \{S\}$.

(e) Accepting sink states can be removed from sets: If $S \in G$ and there is a sink state $q \in S \cap F$, then $G \sim (G \setminus \{S\}) \cup \{S \setminus \{q\}\}$.

Proof. (a) This follows from the definition of $\delta_W$ in the powerset construction.

(b) Note first that $\delta_W(\{S\}, w) = \delta'(S, w)$ in the notation of Definition 4. By item (a) it is enough to consider the case $|w| = 1$. For $T \subseteq Q$ and $f : T \to \{0, 1\}$, denote $\delta(T, f) = \{\delta(q, f(q)) \mid q \in T\}$. For $w = A$ we have

$$\delta'(S \cup R, A) = \{\delta(S \cup R, f) \mid f : S \cup R \to \{0, 1\}\} = \{\delta(S, f) \cup \delta(R, g) \mid f : S \to \{0, 1\}, g : R \to \{0, 1\}\} = \{T \cup V \mid T \in \delta'(S, A), V \in \delta'(R, A)\}.$$

For $w = B$ we have $\delta'(S \cup R, B) = \{\delta(q, b) \mid q \in S \cup R, b \in \{0, 1\}\} = \{\delta(q, b) \mid q \in S, b \in \{0, 1\}\} \cup \{\delta(q, b) \mid q \in R, b \in \{0, 1\}\} = \delta'(S, B) \cup \delta'(R, B)$.

(c) Denote $H = G \setminus \{R\}$ and $T = R \setminus S$, so that $G = H \cup \{S \cup T\}$ and the unions are disjoint. By items (a) and (b) we have

$$\delta_W(G, w) = \delta_W(H, w) \cup \{U \cup V \mid U \in \delta_W(\{S\}, w), V \in \delta_W(\{T\}, w)\}.$$

If $\delta_W(H, w)$ is accepting, then some $X \in \delta_W(H, w)$ satisfies $X \subseteq F$, and $\delta_W(G, w)$ is also accepting. Conversely, if $\delta_W(G, w)$ is accepting, then some $X \in \delta_W(G, w)$ satisfies $X \subseteq F$. If $X \in \delta_W(H, w)$, then we are done; otherwise $X = U \cup V$ with $U \in \delta_W(\{S\}, w)$ and $V \in \delta_W(\{T\}, w)$. Then $U \subseteq F$, and since $S \subseteq H$, the game state $\delta_W(H, w)$ is accepting.
(d) The game state $\delta_W(\{S\}, w)$ is not accepting for any $w \in \{A, B\}^*$, since each of its elements contains $\delta(q, v) \notin F$ for some $v \in \{0, 1\}^{|w|}$. Thus $\delta_W(G, w) = \delta_W(G \setminus \{S\}, w) \cup \delta_W(\{S\}, w)$ is accepting if and only if $\delta_W(G \setminus \{S\}, w)$ is.

(e) Denote $R = S \setminus \{q\}$ and let $w \in \{A, B\}^*$ be arbitrary. Since $q$ is a sink state, $\delta_W(\{\{q\}\}, w) = \{\{q\}\}$. By item (b), $\delta_W(\{S\}, w) = \{T \cup V | T \in \delta_W(\{R\}, w), V \in \delta_W(\{\{q\}\}, w)\} = \{T \cup \{q\} | T \in \delta_W(\{R\}, w)\}$. Since $q \in F$, this game state is accepting if and only if $\delta_W(\{R\}, w)$ is. Thus $\{S\} \sim \{R\}$, and item (a) implies $G \sim (G \setminus \{S\}) \cup \{R\}$.

Define a partial order on $2^2^Q$ as follows: $G \leq H$ if for each $S \in G$ there exists $R \in H$ with $R \subseteq S$. By Lemma 7(a) and (b), the transition function $\delta_W$ is monotonic with respect to $\leq$ in the sense that $G \leq H$ implies $\delta_W(G, w) \leq \delta_W(H, w)$ for all $w \in \{A, B\}^*$. Furthermore, if $G \leq H$ and $G$ is accepting, then some $S \in G$ and $R \in H$ satisfy $R \subseteq S \subseteq F$, hence $H$ is also accepting.

The next lemma helps prove equivalences of game states and words.

**Lemma 8.** Recall the assumptions of Lemma 7

(a) If $G \leq H$ and $H \leq G$, then $G \sim H$.

(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 $v \equiv_W(\mathcal{L}(A)) w$.

**Proof.** (a) This is immediate from the monotonicity of $\delta_W$.

(b) Let $G \in 2^2^Q$ be a game state and suppose $\delta_W(G, v)$ is accepting, so there exists $P \in \delta_W(G, v)$ consisting of accepting states of $A$. We claim $\delta_W(G, w)$ is also accepting. We have $\delta_W(G, v) = \bigcup_{S \in G} \delta_W(\{S\}, v)$ by definition, and similarly for $w$, so we may assume $G = \{S\}$ is a singleton. By Lemma 7(b), for each $q \in S$ there exists $R_q \in \delta_W(\{\{q\}\}, v)$ such that $P = \bigcup_{R_q \subseteq F} R_q$. In particular $R_q \subseteq F$, so each $\delta_W(\{\{q\}\}, v)$ is accepting. By assumption $\delta_W(\{\{q\}\}, w)$ is also accepting, so there exists $T_q \in \delta_W(\{\{q\}\}, w)$ with $T_q \subseteq F$. By Lemma 7(b), we have $\bigcup_{q \in S} T_q \in \delta_W(\{S\}, w)$, so $\delta_W(\{S\}, w)$ is accepting. This shows $v \equiv_W(\mathcal{L}(A)) w$.

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: $a^n < D(n) < b^n$ holds for large enough $n$ if $a < 2 < b$. This follows from $\binom{n}{(n/2)} \leq \log_2 D(n) \leq (1 + O(\log n/n))(\binom{n}{(n/2)})$ and the well known asymptotic formula $\binom{n}{(n/2)} = \Theta(2^{n^2/\sqrt{n}})$.

**Proposition 9.** Let $A$ an $n$-state DFA. The number of states in the minimal DFA for $W(\mathcal{L}(A))$ is at most the Dedekind number $D(n)$.

**Proof.** Every game state is equivalent to an antichain by Lemma 7(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 10 (Subset factory gadget).** Let \(GenSubset\) be the graph in Figure 2. For \(i \in \{1, \ldots, n\}\), denote \(o_i = e_{2n+i-1}\) (the \(n\) rightmost states labeled by \(e\)). For \(S \subseteq \{1, \ldots, n\}\), let \(w^S\) be the concatenation \(w_1w_2\ldots w_n\) where \(w_i = BA\) if \(i \in S\), and \(w_i = AB\) if \(i \notin S\). Then \(\delta_W(\{\{b_1\}\}, w^S) \sim \{\{o_i \mid i \in S\}\}\) for each binary DFA that contains \(GenSubset\) 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.

![Figure 2: GenSubset\(_n\), the subset factory gadget.](image)

**Proof.** Denote \(f_i = e_{3i-2}\). For \(i \in \{1, \ldots, n\}\) and \(S \subseteq \{1, \ldots, i-1\}\), denote \(S_i = \{e_{2i-4+j} \mid j \in S\}\). Consider the game state \(H(i, S) = \{\{b_i\} \cup S_i\}\). If the automaton reads \(AB\), the resulting game state is

\[\delta_W(H(i, S), AB) = \{\{s_i, e_{3i}\} \cup S_{i+1}, \{b_{i+1}\} \cup S_{i+1}\} \sim H(i+1, S)\.

In the case of \(BA\) we instead have

\[\delta_W(H(i, S), BA) = \{\{b_{i+1}, s_i\} \cup S_{i+1}, \{b_{i+1}, f_i\} \cup S_{i+1}\} \sim H(i+1, S \cup \{i\})\.

7
In both cases the final steps follow from Lemma 7[d]. By Lemma 7[e] we also have $H(n + 1, S) \sim \{S_{n+1}\}$ since $b_{n+1}$ is an accepting sink state.

For $i = n$ we now have $\delta_W(\{\{b_i\}\}, w_S^{\text{gen}}) = \{S_{n+1}\} = \{\{a_i | i \in S\}\}$. □

**Lemma 11** (Game state factory gadget). Let $\text{GenState}_n$ be the graph in Figure 3 and $A$ any DFA over $\{0, 1\}$ that contains it. For all $G = \{S_1, \ldots, S_l\}$ where each $S_i \subseteq \{r_1, \ldots, r_n\}$, let $w_G^{\text{gen}} \in \{A, B\}^{(3n+1)}$ be the concatenation of $A w_S^{\text{gen}}, A^n$ for $i \in \{0, \ldots, \ell\}$. Then $\delta_W(\{\{a_i\}\}, w_G^{\text{gen}}) \sim G \cup \{\{a_i\}\} \cup G'$ for some game state $G'$ that does not contain a subset of the states of $\text{GenState}_n$.

![Figure 3: GenState_n, the game state factory gadget.](image)

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 $w_S^{\text{gen}}$ 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 $a_i$.

**Proof.** Suppose we have reached a game state of the form $H = \{\{a_1\}, \{r_i | i \in S_1\}, \ldots, \{r_i | i \in S_k\}\} \cup G'$ where $G'$ does not contain any subset of $\text{GenState}_n$. We prove that by reading $A w_S^{\text{gen}}, A^n$, we reach a game state of the form $\{\{a_1\}, \{r_i | i \in S_1\}, \ldots, \{r_i | i \in S_k\}\} \cup G''$. We analyze the elements of $H$ separately.

- Because $|A w_S^{\text{gen}}, A^n| = 3n + 1$ is the size of the rightmost cycle, we have $\delta_W(\{\{b_i\}\}, Aw_S^{\text{gen}}, A^n) \sim \{\{r_i | i \in S_j\}\} \cup G''$ for each $j \leq k$, where each set in $G''$ contains a state outside the gadget.
- The game state $\{\{a_1\}\}$ first evolves into $\delta_W(\{\{a_1\}\}, A) = \{\{a_2\}, \{b_1\}\}$. The component $\{\{a_2\}\}$ becomes $\{\{a_1\}\}$ when we read $w_{S_{k+1}}^{\text{gen}}$. As for $\{\{b_1\}\}$, Lemma 11 gives $\delta_W(\{\{b_1\}\}, w_{S_k}^{\text{gen}}) = \{\{a_i | i \in S_k\}\}$, and then $\delta_W(\{\{a_i | i \in S_{k+1}\}\}, A^n) = \{\{r_i | i \in S_{k+1}\}\} \cup G''$ where every set in $G''$ contains a state outside of $\text{GenState}_n$.
- The game state $G'$ evolves into some $G''$ each of whose sets contains a state not in $\text{GenState}_n$, since the gadget cannot be re-entered.

By Lemma 7[a] we have $\delta_W(H, A w_S^{\text{gen}}, A^n) \sim \{\{a_1\}, \{r_i | i \in S_1\}, \ldots, \{r_i | i \in S_k\}\} \cup G'' \cup G'' \cup \bigcup G_j$. We obtain $w_G^{\text{gen}}$ as the concatenation of these words. □
Lemma 12 (Testing gadget). Let Testing\(_n\) be the graph in Figure 4.

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

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

![Figure 4: Testing\(_n\), the testing gadget.](image)

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 \(r\).

Proof. (a) For \(I \subseteq \{1, \ldots, 2n\}\), denote \(S_I = \{q_i \mid i \in I\}\). If \(2n \notin I\), let \(J = \{i+1 \mid i \in I\}\). A simple case analysis together with Lemma 7(d) and (e) shows that \(\delta_W(S_I, A) \sim \{S_J\}\) and \(\delta_W(S_I, B) = \begin{cases} \emptyset & \text{if } n \in I, \\ \{S_J\} & \text{otherwise.} \end{cases}\)

Then \(G = \delta_W(S_I, w_P^{\text{test}})\) is accepting if and only if \(G \sim \{\{q_i+n \mid i \in I\}\}\). This is equivalent to \(w[n-i+1] = A\) for all \(i \in I\), i.e. \(I \subseteq P\).

(b) If \(G \in 2^V\) and \(w\) with \(|w| \geq 2n\), then every \(S \in \delta_W(G, w)\) satisfies \(S \subseteq \{r, r'\}\).

Theorem 13. For each \(n > 0\) there exists a DFA \(A_n\) over \(\{0, 1\}\) with \(15n + 3\) states such that the minimal DFA for \(W(L(A_n))\) has a least \(D(n)\) states.

Together with Proposition 9 this implies that the state complexity of \(W\) restricted to regular languages grows doubly exponentially.

Proof. Let \(A_n\) 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 \( X_G = \delta_W (\{ \{ a_1 \} \}, w_G^{\infty} r) \). Lemma 11 gives \( X_G \sim \{ \{ a_1 \} \} \cup G \cup G' \) 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 \( X'_G = \delta_W (X_G, AB^{2n}A^{2n+1}w_P^{\text{test}}) \). We claim that \( X'_G \) is accepting iff some element of \( G \) is a subset of \( \{ r_i \mid i \in P \} \). By Lemma 11 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 \( B_8 \), 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 \( \delta_W (G, AB^{2n}A^{2n+1}) \) consists of the sets \( \{ q_i \mid r_i \in S \} \) for \( S \in G \), 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 12 the former sets produce an accepting game state in \( X'_G \) iff some \( S \in G \) is a subset of \( \{ r_i \mid i \in P \} \).

We have found \( D(n) \) nonequivalent states in \( W(A) \).

## 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, which is lower than the one proved for unrestricted regular languages, but still superexponential. We do not know if the true state complexity is superexponential for this class. Our proof technique is based on tracing the evolution of individual states of a DFA \( A \) in the winning set automaton \( W(A) \) when reading several \( A \)-symbols in a row.

Our motivation for studying bounded regular languages comes from the fact that they 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. Recall that the definition of winning sets in [13] was motivated by the study of entropy of shift spaces. Since the winning set operation preserves entropy, in particular it preserves the property of having entropy zero. See Section 4 of [11] for discussion on zero entropy sofic shifts and their structure. Analogously, since bounded regular languages are exactly those for which the number of words of given length grows polynomially, and the winning set operation preserves this number by Proposition 3, the winning set of a binary bounded language is also bounded.

**Definition 14 (Histories of Game States).** Let \( A = (Q, \{ 0, 1 \}, q_0, \delta, F) \) be a DFA. Let \( G \in 2^{2^2} \) be a game state of \( W(A) \), and for each \( i \geq 0 \), let \( G_i \sim \delta_W (G, A') \) be the game state with all supersets removed as per Lemma 11. A history function for \( G \) is a function \( h \) that associates to each \( i > 0 \) and each set \( S \in G_i \), a parent set \( h(i, S) \in G_{i-1} \), 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 \( S \in G_i \) 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 \( S \in G_i \) has at least one set \( R \in G_{i-1} \) 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. 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 \( A = (Q, \{0,1\}, q_0, \delta, F) \) that recognizes a bounded binary language and has disjoint cycles. Let the lengths of the cycles be \( k_1, \ldots, k_p \), and let \( \ell \) be the number of states not part of any cycle.

We define a preorder \( \preceq \) on the state set \( Q \) by reachability: \( p \preceq q \) holds if and only if there is a path from \( p \) to \( q \) in \( A \). For two history functions \( h, h' \) of a game state \( G \), we write \( h \preceq h' \) if for each \( i > 0 \), each \( S \in G_i \) and each \( q \in S \), there exists a function \( f : h(i,S,q) \rightarrow h'(i,S,q) \) with \( p \preceq f(p) \) 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 \( h \preceq h' \) and \( h' \not\preceq h \). 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 \( A \) 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 15.** Let \( G \in 2^{Q^0} \) be any game state of \( W(A) \). Then there exist \( k \leq \text{lcm}(k_1, \ldots, k_p) + 2n + \max_{x \neq y} \text{lcm}(k_x, k_y) \) and \( m \leq \text{lcm}(k_1, \ldots, k_p) \) such that \( \delta_W(G,A^k) \sim \delta_W(G,A^{k+m}) \).

The idea of the proof is that under a minimal history function, no state \( q \in S \in G \) 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.

**Proof.** Denote the cycles of \( A \) by \( C_1, \ldots, C_p \), so that \( |C_i| = k_i \) for each \( i \). Let \( h \) be a minimal history function of \( G \). Define \( G_i \) for \( i \geq 0 \) as in Definition 14.

Let \( t \geq 0 \), \( S \in G_t \), and \( q \in S \) be arbitrary, and let \( S_0, \ldots, S_t = S \) and \( q_0, \ldots, q_t = q \) be their histories under \( h \). The history of \( q \) travels through some of the cycles of \( A \), never entering the same cycle twice. We split the sequence \( q_0, \ldots, q_n \) into words over \( Q \) as \( u_0v_1^{p_1}u_1v_2^{p_2}u_2 \cdots v_n^{p_n}u_r \), where

- each \( p_j \geq 1 \),
- each \( v_j \) consists of the states of some cycle, which we may assume is \( C_j \), repeated exactly once,
- the \( u_j \) do not repeat states and each \( u_j \) does not contain any states from \( C_{j+1} \).

Intuitively, \( v_j \) represents a phase of the history where the state stays in a cycle for several loops, and the \( u_j \) represent transitions from one loop to another. Each \( u_j \) ends right before the time step when the history of \( q \) enters the loop \( C_{j+1} \). It may share a nonempty prefix with \( v_j \). See Figure 5(a).
Figure 5: (a) Definition of the words $v_{i-1}$, $u_{i-1}$ and $v_i$. (b) A part of the history $h(\cdot,S,q)$ (above) and $h'(\cdot,S,q)$ (below). The braces count the number of repetitions of $v_{i-1}$ and $v_i$. 
We claim that \( p_i k_i \leq \max_{x \neq y} \text{lcm}(k_x, k_y) \) holds for all \( 1 \leq i \leq r \). Assume the contrary. Since \( k_i = |v_i| \), we have in particular \( p_i |v_i| > \text{lcm}(|v_{i-1}|, |v_i|) \) for some \( i \), so that \( a[v_{i-1}] = b[v_i] \) holds for some \( a > 0 \) and \( 0 < b \leq p_i \). Denote \( s = |u_0 v_1^{p_1} \cdots u_{i-2}| \), which is the time step after which the history of \( q \) enters the repetitive portion of the previous loop \( C_{i-2} \). Denote \( K = |v_{i-1}^{p_{i-1}} a u_{i-1}| \) and \( q_{s+1}, q_{s+2}, \ldots, q_{s+K} = v_{i-1}^{p_{i-1}} a u_{i-1} b \). Note that \( q_{s+1}, q_{s+2}, \ldots, q_{s+K} = v_{i-1}^{p_{i-1}} a u_{i-1} b \), so that we have \( q_{s+i} \leq q_{s+j} \) for all \( 1 \leq j < K \) and \( q_{s+K} < q_{s+K} \). See Figure 3(b). Our goal is to modify the history function \( h \) by replacing each \( q_{s+i} \) with \( q_{s+i} \) in this range.

In \( A \) we have transitions from \( q_s \) to both \( q_{s+1} \) and \( q_{s+1} \), and from each \( q_{s+j} \) to \( q_{s+j+1} \), as well as from \( q_{s+K} \) to \( q_{s+K+1} \). For \( q_i \leq q \) the game state \( \delta_W(G, A^{s+j}) \) contains \( S_{s+j} = (S_{s+j} \setminus \{q_j\}) \cup \{q_{s+j}\} \). We also have \( S_{s+K+1} \in \delta_W(S_{s+K}, A) \). There are now two possibilities. If \( q_{s+K} \in S_{s+K} \), then \( S_{s+K} \in \delta_W(G, A^{s+K}) \) is a proper subset of \( S_{s+K} \), which contradicts our choice of \( G_{s+K} \) as a version of \( \delta_W(G, A^{s+K}) \) with all proper supersets removed.

If \( q_{s+K} \notin S_{s+K} \), then we may define a new history function \( h' \) by defining \( h'(s+K+1, S) = S_{s+K}, h'(s+K+1, S, q) = h(s+K+1, S, q) \), and \( h'(t, S', q') = h(t, S', q') \) for all other choices of \( t, S' \) and \( q' \). Then the function \( f : h(s+K+1, t, S, q) \) shows \( h' \) is minimal, which contradicts our choice of \( G_{s+K} \) as a version of \( \delta_W(G, A^{s+K}) \) with all proper supersets removed.

Denote \( L = \text{lcm}(k_1, \ldots, k_p) \). If \( t \geq \ell + 2 \ell + p \cdot \max_{x \neq y} \text{lcm}(k_x, k_y) \), then \( p_i k_i = L + \ell \), which implies \( q_i = q_{\ell+L} \) (note that \( |u_0| \leq \ell \), so that \( q_i, q_{\ell+L} \in C_1 \). Since this holds for every history of every state of \( S \) under \( h \) and each state of each set \( S_t \) for \( i \leq t \) can be chosen as \( q_i \) for some \( q \in S \), we have \( S_t = S_{\ell+L} \). Then \( S \in \delta_W(S_{\ell+L}, A^{t-\ell-L}) = \delta_W(S_{\ell+L}, A^{t-\ell-L}) \), so in particular \( S \in \delta_W(G, A^{t-L}) \sim G_{t-L} \). On the other hand, \( S \in \delta_W(S_{\ell+L}, A^{t-\ell-L}) = \delta_W(S_{t-L}, A^{t-\ell-L}) \), so \( S \in \delta_W(G, A^{t-L}) \sim G_{t-L} \). Since \( S \in G_t \) was arbitrary, we have \( G_t \sim G' \sim G'' \sim G_{t+L} \) for some game states \( G', G'' \sim 2^{2^q} \). By considering \( t+L \) instead of \( t \) and doing the same analysis, we obtain \( G_t \sim G_{t+L} \).

**Theorem 16.** Let \( A \) 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 \( W(L(A)) \) has at most \( \sum_{m=0}^{t+1} (p \cdot \max_{x \neq y} \text{lcm}(k_x, k_y) + 2\ell + 2\text{lcm}(k_1, \ldots, k_p)) \) states.

**Proof.** Denote the minimal DFA for \( W(L(A)) \) by \( B \). We may assume that \( A \) is minimal, and then it has disjoint cycles, as otherwise the number of length- \( n \) words in \( L(A) \) 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 \( \ell \) the number of remaining states, and denote \( P = p \cdot \max_{x \neq y} \text{lcm}(k_x, k_y) + 2\ell + 2\text{lcm}(k_1, \ldots, k_p) \). Then any \( w \in L(W(A)) \) has \( |w_B| \leq \ell + p \), as otherwise Bob can win by choosing to leave a cycle whenever possible.

Consider a word \( w = A^{\ell_0} B A^{\ell_1} B \cdots B A^{\ell_m} \) with \( 0 \leq m \leq \ell + p \). If \( \ell_i \geq P \) for some \( i \), then Lemma 14 implies \( \delta_W(G, A^{\ell_i}) \sim \delta_W(G, A^{\ell_i}) \) for the game state \( G = \delta_W(\{v_0\}, A^{\ell_0} B \cdots B A^{\ell_m}) \) and some \( t < \ell_i \). Thus the number of distinct states of \( B \) 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 GenSubset$_n$ and 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 $A = (Q, \{0, 1\}, q_0, \delta, F)$ 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 6 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 $W(L(A))$ so the results of this section apply to every DFA with the structure of a 1-bounded chain DFA.

![Figure 6: A 1-bounded chain DFA. Any states except $n-1$ can be final.](image)

**Lemma 17.** Let $A$ be an $n$-state 1-bounded chain DFA. Let $\equiv$ stand for $\equiv_{W(L(A))}$.

(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 8 simplifies the analysis.

**Proof.** Label the states of $A$ by $\{0, 1, \ldots, m-1\}$ as in the definition of 1-bounded chain DFA. For (a) [(b)] and [(c)] we “unroll” the self-loop at $m-1$ labeled by 1 to obtain an equivalent automaton with an infinite chain of states, simplifying the arguments. We also argue in terms of the NFA for $W(L(A))$, saying that “we produce a set $R \subseteq Q$ from $S \subseteq Q$ by reading $w \in \{A, B\}^*$ if $R \in \delta_W(\{S\}, w)$.

In this formalism (a) means that for any set produced from $\{q\}$ by reading $AB$, we can produce a subset of it by reading $BA$. To see what sets can be produced, we use a spacetime diagram (Figure 7) where the time increases to
the south. In this diagram, by reading a $B$, a selected state will spread south and southeast. By reading an $A$, we have both possibilities, resulting in multiple sets. The claim follows directly.

![Diagram](image)

Figure 7: Sets produced with $AB$ and $BA$. The arrows indicate a subset relation.

We now prove (b). By Lemma 8(b) it is enough to consider singleton states $\{q\}$, and without loss of generality we assume $q = 0$. By item (a) of this result and Lemma 8(a) it is sufficient to prove that for every set obtained from $B^{k+1}A^kB^k$, we can produce a subset of it by reading $B^kA^kB^{k+1}$. After reading $B^{k+1}$, we have the interval $\{0, \ldots, k + 1\}$. After $A^k$ we get a set included in $\{0, \ldots, 2k + 2\}$ where the distance between every two consecutive elements is less than $k$. After reading $B^k$ the gaps are filled and we get an interval containing $\{k, \ldots, 2k + 2\}$. When reading $B^kA^kB^{k+1}$ we do the following: get $\{0, \ldots k\}$ with $B^k$, make every element go to position $k$ with $A^k$, and extend with $B^{k+1}$ to get the interval $\{k, \ldots, 2k + 2\}$.

For (c), for the same reason as previously it is sufficient to prove that for every set obtained from $\{0\}$ by reading $A^kB^kB^{k+1}$, we can produce a subset by reading $A^{k+1}B^kB^k$. First we prove that by reading $A^{k+1}B^kB^k$ we can get any singleton set $\{k\}, \{k + 1\}, \ldots, \{2k + 1\}$: By reading $A^{k+1}B^k$, we can get any singleton set between 0 and $k + 1$ and expand it to have any interval of length $k + 1$ between 0 and $2k + 1$. Then by reading $A^k$ we can have a singleton state at the end position of the interval, that is between $k$ and $2k + 1$. See Figure 8.

Now we prove that every set obtained by reading $A^kB^kB^{k+1}$ has at least one of $k, \ldots, 2k + 1$. After reading $A^kB^k$ we have any interval of size $k + 1$ between 0 and $2k$. The state $k$ is always in the interval, and must go somewhere between positions $k$ and $2k + 1$ after $A^{k+1}$.

As for (d), reading $A^{n-1}$ or $A^n$ in any singleton game state $\{q\}$ produces exactly the states $\{q\}, \{q + 1\}, \ldots, \{m - 1\}$ and the loop $\{m\}, \ldots, \{m + p - 1\}$. Likewise, reading $B^{n-1}$ or $B^n$ produces $\{q, q + 1, \ldots, m - 1\} \cup \{m, \ldots, m + p - 1\}$.

Theorem 18. Let $A$ be a 1-bounded chain DFA with $n$ states. The number of states in the minimal DFA of $W(\mathcal{L}(A))$ is $O(n^{1/5}e^{4\pi\sqrt{n}})$. 

15
Figure 8: The sets produced by $A^{k+1}B^kA^k$.

**Proof.** Since $A$ does not accept any word with $n$ or more 1-symbols, $W(L(A))$ contains no word with $n$ or more $B$-symbols. Lemma 14(b) and (c) allow us to rewrite every word of $W(L(A))$ in the form $A^{n_1}B^{n_2}A^{n_3}B^{n_4} \ldots A^{n_{2r-1}}B^{n_{2r}}$, 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 14(d) we can also guarantee $n_1, n_3, \ldots, n_{2r-1} < n$, so that $\sum n_i < 4n$. In [3], 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 $W(L(A))$, and the number of states in its minimal DFA, is at most $1 + \sum_{m=0}^{4n-1} Q(m) = O(n^{1/5}e^{4\pi\sqrt{n/3}})$.

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 $A$ is the minimal automaton for $L$, described in Figure 9. For $S \subseteq Q$, we denote $\overline{S} = \{ \min(S), \min(S) + 1, \ldots, \max(S) \}$, and for any game state $G$ of $W(A)$, denote $\overline{G} = \{ \overline{S} \mid S \in G \}$.

Figure 9: The minimal DFA for $L = (0^*1)^n0^*$. 

16
Lemma 19. Each game state $G$ of $W(A)$ is equivalent to $\overline{G}$.

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.

Proof. We prove by induction that for every $w \in \{A, B\}^*$, the game state $\delta_W(G, w)$ is final iff $\delta_W(\overline{G}, w)$ is.

- For the empty word $\lambda$, $\delta_W(G, \lambda) = G$ is final iff $\{n\} \in G$ iff $\{n\} \in \overline{G} = \delta_W(\overline{G}, \lambda)$.
- For $w = Bv$, it's easy to see that $\delta_W(\overline{G}, B) = \overline{\delta_W(G, B)}$. By the induction hypothesis, $\delta_W(G, Bv) = \delta(\delta_W(G, B), v)$ is final iff $\delta_W(\overline{\delta_W(G, B)}, v) = \delta_W(\overline{G}, Bv)$ is.
- For $w = Av$, we have $\delta_W(\overline{G}, A) = \overline{\delta_W(G, A)}$: For each set $S \in G$, we focus on a set $R$ that can be obtained by a combination of choices by reading one $A$. The set $\overline{R}$ is only defined by the leftmost and rightmost elements in $R$. From $\overline{G}$, by making the same choices for the leftmost and rightmost elements we can produce the same leftmost and rightmost elements as in $R$ to obtain a set equivalent to $\overline{R}$.

By the induction hypothesis, the game state $\delta_W(G, Av) = \delta_W(\delta_W(G, A), v)$ is final iff $\delta_W(\overline{\delta_W(G, A)}, v) = \delta_W(\overline{G}, Av)$ is. Again by the induction hypothesis, this is equivalent to $\delta_W(\overline{G}, A), v) = \delta_W(\overline{G}, Av)$ being final.

Lemma 20. Let $T$ be the set of integer triples $(i, \ell, N)$ 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

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

(a) Each reachable game state of $W(A)$ is equivalent to some $G(i, \ell, N)$ for $(i, \ell, N) \in T$, or to $\emptyset$.

(b) The game states $G(i, \ell, N)$ for $(i, \ell, N) \in T$ are nonequivalent.

(c) Every $G(i, \ell, N)$ for $(i, \ell, N) \in T$ is equivalent to some reachable game state.

The game state $G(i, \ell, N)$ is an interval of intervals, where $i$ is the leftmost position of the first interval, $\ell$ 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.

Proof. We first make the following remarks which follow from Lemma 19. Let $S = \{i, \ldots, j\}$. If $i = j$, then $\delta_W(S, A) \sim \{i\}$, $\{i+1\}$, otherwise $\delta_W(S, A) \sim \{\{i+1, \ldots, j\}\}$. In both cases we also have $\delta_W(S, B) \sim \{\{i, \ldots, j+1\}\}$.

We prove $[\text{a}]$. From $G(i, \ell, N)$, if we read $B$, we have $G(i, \ell + 1, N)$. If we read $A$ and $\ell = 1$, we get $G(i, 1, N + 1)$. If we read $A$ and $\ell > 1$, we obtain $G(i+1, \ell - 1, N)$. The claim follows since the initial game state is $G(0, 1, 1)$, and if a set in the game state contains the state $n + 1$, it is equivalent to $\emptyset$.
For [b] we first distinguish game states with different lengths \( \ell < \ell' \). By reading \( A^k(BA)^k \) for a suitable \( k \geq 0 \) we reach a final game state from \( G(i, \ell, N) \) but not from any \( G(i', \ell', N') \) with \( \ell < \ell' \).

Now we suppose we have game states \( G(i, \ell, N) \) and \( G(i', \ell, N') \). By reading \( A^{\ell-1} \), we get respectively \( G(i + \ell - 1, 1, N) \) and \( G(i' + \ell - 1, 1, N') \), which are intervals of singleton sets. Since these distributions of singleton sets are different, we can read \( (BA)^k \) for a suitable \( k \geq 0 \) to obtain a final game state from one of them but not the other.

For [c] reading \( (BA)^k A^{N-1} B^{\ell-1} \) leads to a game state equivalent to \( G(i, \ell, N) \).

In particular, the minimal DFA for \( W(L) \) has \( |T| + 1 \) states, and counting them yields the following.

**Proposition 21.** The minimal DFA for \( W(L) \) has \( n^3 \) + \( n^2 + \frac{11n}{2} + 2 \) states.

**Proposition 22.** \( W(L) \) 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.** Every \( w \in W(L) \) satisfies \( |w|_A \geq n \) since Bob can choose to play only 0s, and \( |w|_B \leq n \) since he can choose to play only 1s. To see that the suffix condition is necessary, note first that only game states of the form \( G(i, 1, N) \) can be accepting. Since reading \( A \) decreases the parameter \( \ell \) by one, and \( B \) increases \( \ell \) 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 Membership and Intersection

In this section we investigate the computational complexity of deciding membership in the winning set of a regular language, and nonemptiness of intersection with a given regular language over \( \{A, B\} \). The former turns out to be P-complete and the latter PSPACE-complete. Our hardness results rely on simulation of boolean circuits. Such simulation allows us to reduce P-hardness of circuit evaluation and PSPACE-hardness of the iterated circuit prediction to the aforementioned decision problems on the winning set.

For the purposes of this section, a **boolean circuit** is a finite directed acyclic graph \( C \). There are five types of nodes in \( C \): **AND**-gates and **OR**-gates with indegree 2 and arbitrary outdegree, **NOT**-gates of indegree 1 and arbitrary outdegree, input nodes with indegree 0 and outdegree 1, and output nodes with indegree 1 and outdegree 0. If \( C \) contains \( k \) input and \( m \) output nodes, it transforms elements of \( \{T, F\}^k \) into those of \( \{T, F\}^m \) by evaluation of the gates. In the **circuit value problem**, we are given a circuit \( C \) with \( k \) inputs and a single output, and a vector \( a \in \{T, F\}^k \), and are asked whether \( C(a) = T \); this problem is P-complete with respect to logarithmic space reductions [2 Theorem 6.30]. In the **iterated circuit problem**, we are given a circuit \( C \) with \( k \) inputs and outputs, a vector \( a \in \{T, F\}^k \) and an index \( i \in \{0, \ldots, k-1\} \), and are asked whether \( C^n(a)_i = T \) for some \( n \geq 0 \); this problem is PSPACE-complete with respect to polynomial time reductions [5 Lemma 3].
The general idea is that each gate of the circuit corresponds to a gadget formed by a subset of states in the automaton, and the simulation is done “backwards in time” in the sense that the initial state of the winning set automaton corresponds to the output of the circuit. Each output node of the circuit corresponds to a pair of states in the automaton, and an output vector is encoded by choosing one state from each pair as initial. Then by reading an appropriate word, we can construct a game state corresponding to all possible inputs that map to the chosen output, and accept only if the chosen input is represented in this game state. The following result codifies this idea.

**Lemma 23.** Let $C$ be a boolean circuit with $k$ inputs, $m$ outputs, and $n$ gates. Then there is a number $p = O(n)$ and an acyclic DFA $A_C = (Q, \{0, 1\}, q_0, \delta, F)$ with $O(n(m + k + n))$ states, and special states $q_i^T, q_i^F \in Q$ for $i = 0, \ldots, k − 1$ and $p_i^T, p_i^F \in Q$ for $i = 0, \ldots, m − 1$ such that the following holds in $W(A_C)$: for all $a \in \{T, F\}^n$ we have $\delta_W(\{p_0^a, \ldots, p_{m−1}^a\}, (AAB)^a) = G \cup H$, where $G = \{\{q_0^{b_0}, \ldots, q_{k−1}^{b_{k−1}}\} \cup C(b_0, \ldots, b_{k−1}) = a_0, \ldots, a_{m−1}\}$ and every $X \in H$ satisfies $(q_i^T, q_i^F) \subseteq X$ for some $i$.

Here, $G$ encodes exactly the inputs of the circuit that produce the output $a$, while $H$ contains artifacts of the construction that we can safely ignore later on.

**Proof.** We implement gadgets for OR-gates and NOT-gates of outdegree 1, and SPLIT-gates which split one input into multiple copies. They are shown in Figure 10. Then AND-gates can be implemented by de Morgan’s law, and gates with arbitrary outdegree by SPLIT-gates. The boolean values are represented in a dual-rail fashion, so a gadget corresponding to a gate $g$ with indegree $n_1$ and outdegree $n_2$ will have $2n_1$ input states $s_i^b = s_i^b(g)$ for $b \in \{T, F\}$ and $i = 0, \ldots, n_1 − 1$, and $2n_2$ output states $t_i^b = t_i^b(g)$ for $b \in \{T, F\}$ and $i = 0, \ldots, n_2 − 1$. A subset of the $q_i^b$ is consistent if it contains exactly one of $q_i^T$ and $q_i^F$ for each $i$, and excessive if it contains both $q_i^T$ and $q_i^F$ for some $i$. Similar definitions are given for the output states. The gates are constructed so that by reading $AAB$, a consistent subset $S$ of the output states is transformed into a set of subsets of the input variables whose consistent members are exactly the inputs corresponding to $S$, and the rest are excessive.

![Figure 10: Implementation of the gate gadgets: OR-gate on the left, NOT-gate in the middle, and SPLIT-gate of outdegree $n$ on the right.](image-url)
For the OR-gate, the set $S = \{t_1^T\}$ is transformed by $AA$ into three sets that are expanded with $B$ to obtain the sets $\{s_0^T, s_1^T\}$, $\{s_0^F, s_1^F\}$ and $\{s_0^B, s_1^B\}$. The set $S = \{t_0^T\}$ is transformed by $AAB$ into $\{s_0^F, s_1^F\}$. If $S = \{t_1^T, t_1^F\}$ is excessive, then so is the result.

For the NOT-gate, we simply exchange the roles of the states corresponding to $T$ and $F$. This gadget also preserves excessive sets. The SPLIT-gate gadget transforms a set of the form $S = \{t_0^b, t_1^b, \ldots, t_{n-1}^b\}$ for $b \in \{T,F\}$ into $\{s_0^b\}$, and other consistent or excessive sets into excessive sets.

Given a boolean circuit $C$, we assemble these gadgets into the DFA $A_C$ in the following way, depicted in Figure 11. First, we may assume that each edge on $C$ connects two gates of adjacent depth (minimal distance from an input node), and that each output node has the same depth $d(C)$. If this is not the case, we can enforce it by replacing some subset of edges by paths of SPLIT-gates of degree 1. For the input and output nodes of the circuit, we add the states $q_i^T, q_i^F \in Q$ for $i = 0, \ldots, k - 1$ and $p_i^T, p_i^F \in Q$ for $i = 0, \ldots, m - 1$ to the automaton. Next, we process each gate $g$ in increasing order of depth. Let $n_1, n_2 \geq 1$ be the indegree and outdegree of $g$ in $C$. We add to $Q$ copies of the gadget corresponding to $g$ and a gadget corresponding to a SPLIT-gate $S_g$ of degree $n_2$, and identify $t_0^b(g)$ with $s_0^b(S_g)$ for each $b \in \{T,F\}$. For each incoming edge $h \to g$, let $g$ be the $j$th out-neighbor of $h$ and $h$ the $k$th in-neighbor of $g$. Then we identify $t_1^j(S_h)$ with $s_1^j(g)$ for $h \in \{T,F\}$. If $d = d(C)$ is the depth of $C$, then we do not add the SPLIT-gate gadget $S_g$, but identify the input nodes $t_0^b(g)$ with $p_i^b$ instead, where $g$ is the $i$th output node of $C$. This concludes the construction of $A_C$.

![Figure 11: Example construction of $A_C$, with $C$ on the left.](image)

By the construction, every path from some output state $p_i^b \in Q$ to a given state $q \in Q$ has the same length in $A_C$. For $\ell = 0, \ldots, 2d(C) - 3$, let $S_\ell = \{r_i^b(\ell) \in Q \mid b \in \{T,F\}\}$ be the states for which this length is $3\ell$: they are exactly the input and/or output states of the gadgets used in the construction. We define consistent and excessive subsets of $S_\ell$ as above, and for $a \in \{T,F\}^{n_2}$, define $S_\ell(a) = \{r_i^a \mid i = 0, \ldots, n_2 - 1\}$. For $S \subseteq S_\ell$, Lemma 7(b) implies that $\delta_W(S, AAB)$ contains exactly those subsets of $S_{\ell+1}$.
where each gate gadget is evaluated separately. Thus excessive subsets of $S_t$ are transformed into sets of excessive subsets of $S_{t+1}$, while consistent subsets $S_t(a)$ are transformed into sets of excessive sets and exactly the consistent sets $S_{t+1}(b)$ for those $b \in \{T, F\}^{w+1}$ that produce the vector $a$ when the corresponding gates are applied to them. We can then choose $p = 2d(C) - 3$, and the claim follows.

We also need the fact that the reversal of the winning set of a regular language can be recognized by a DFA of simply exponential size. The construction is similar to the standard method of recognizing the reversed language of an alternating finite automaton \[4\]. The reversal of a language $L$ is denoted $L^R$.

**Definition 24** (Canonical DFA for $W(A)^R$). Let $A = (Q, \{0, 1\}, q_0, \delta, F)$ be a binary DFA. We construct a DFA $A^W = (2^Q, \{A, B\}, q_0^W, \delta^W, F^W)$ as follows. The initial state is $q_0^W = F$, the final states are $F^W = \{S \subseteq Q \mid q_0 \in S\}$, and the transition function is

$$
\delta^W(S, A) = \{q \in Q \mid \delta(q, 0) \in S \text{ or } \delta(q, 1) \in S\},
$$

$$
\delta^W(S, B) = \{q \in Q \mid \delta(q, 0) \in S \text{ and } \delta(q, 1) \in S\}.
$$

**Lemma 25.** In the notation of Definition 24, we have $L(A^W) = W(L(A))^R$.

**Proof.** Let $w \in \{A, B\}^+$. We prove by induction on $|w|$ that $\delta^W(F, w) = S := \{q \in Q \mid w^R \in W(L_q(A))\}$, from which the claim follows. The case $w = \lambda$ is clear. Suppose then $w = vc$ with $c \in \{A, B\}$. Then $\delta^W(F, v) = \{q \in Q \mid v^R \in W(L_q(A))\}$ by the induction hypothesis. If $c = A$, then

$$
\delta^W(F, w) = \delta^W(\delta^W(F, v), A)
$$

$$
= \{q \in Q \mid v^R \in W(L_{\delta(q, 0)}(A)) \cup W(L_{\delta(q, 1)}(A))\}
$$

$$
= \{q \in Q \mid v^R \in A^{-1}W(L_q(A))\} = S.
$$

The analogous equations hold for $c = B$ with union replaced by intersection.

**Theorem 26.** Given a binary DFA $A$ and a word $w \in \{A, B\}^+$, deciding whether $w \in W(L(A))$ is P-complete with respect to logarithmic space reductions.

**Proof.** Let $C$ be a boolean circuit with $k$ input nodes, $x_1, \ldots, x_k$ and one output node $y$, and let $a \in \{T, F\}^k$ be arbitrary. Let $A_C$ be the DFA constructed in Lemma 23 for $C$ and $p$ the corresponding integer. We choose $q_0^T$ as the initial state and $F = \{p_{0}^{a_0}, \ldots, p_{k-1}^{a_{k-1}}\}$ as the final states of $A_C$.

Now the game state $\delta^R(\{q_0^T\}, (AAB)^p)$ of the winning set automaton consists of excessive sets, which are never accepting, and the consistent sets $\{p_{0}^{b_0}, \ldots, p_{k-1}^{b_{k-1}}\}$ for all $b \in \{T, F\}^k$ with $C(b) = T$. Since $F$ is the set of final state, $(AAB)^p \in W(A_C)$ if and only if $C(a) = T$. The transformation of $C$ into $A_C$ in Lemma 23 can be done in logarithmic space. Thus we have a logspace-reduction of the circuit value problem into the winning set membership problem.

To prove that the decision problem is in P, consider from the DFA for $W(L(A))^R$ given in Definition 24. It suffices to read $w^R$ in this automaton one symbol at a time, keeping track of a single set from $2^Q$, which can be done in polynomial time.
Theorem 27. Given a DFA $A$ over $\{0, 1\}$ and an NFA $B$ over $\{A, B\}$, it is PSPACE-complete to decide whether $W(L(A)) \cap L(B) \neq \emptyset$. The same holds if $B$ is assumed deterministic.

Proof. First we prove that the problem is in PSPACE. For this let $A$ be a DFA with $n$ states and $B$ an NFA with $k$ states. The DFA $A^W$ of Definition 24 for $W(L(A))^R$ has $2^n$ states, and reversing the transitions of $B$ results in an NFA for $L(B)^R$ with $k$ states. Their product NFA $P$ recognizes the intersection using $2^n$ states, which can be computed as needed in polynomial space. Therefore by Savitch's accessibility theorem, we can decide if there is a path from the initial state of $P$ to a final state in space $O(\log^2(k2^n)) = O((n + \log(k))^2)$.

Conversely, we claim that given a DFA $A$ and a word $w \in \{A, B\}^*$, deciding whether $W(L(A)) \cap w^* \neq \emptyset$ is PSPACE-hard. Let $C$ a boolean circuit with $n$ gates, $k$ input nodes and $k$ output nodes $x_0, \ldots, x_{k-1}$, and take a word $a \in \{T, F\}^k$ and an index $i \in \{0, 1, \ldots, k - 1\}$. Create a new circuit $C'$ as follows. Add $k$ new output nodes $y_0, \ldots, y_{k-1}$. Add new gates so that $y_j = x_i \lor x_j$ for all $j \in \{0, 1, \ldots, k - 1\}$, so that the $x_i$ are no longer output nodes. Then for all $b \in \{T, F\}^k$, we have $C'(b) = C(b)$ if $C_i(b) = F$, and $C'(b) = T^k$ otherwise.

Let the DFA $A_{C'}$ and $p = O(n)$ be given by Lemma 23 for the circuit $C'$, and construct a new DFA $B_{C'}$ from it as follows. Merge the two states $q_i^b$ and $p_i^b$ for each $i \in \{0, 1, \ldots, k - 1\}$ and $b \in \{T, F\}$. Add a new initial state $q_0$ and a gadget in the shape of a tree whose root is $q_0$ and whose leaves are the states $q_i^T$ similar to the left half of the OR-gate gadget in Figure 10, so that $\delta_w(\{q_0\}, (AAB)^{\log(k)}) = \{q_0^0, \ldots, q_{k-1}^0\}$. Choose $F = \{q_0^0, \ldots, q_{k-1}^0\}$ for the final states of $B$. For each $t \geq 0$ we now have $(AAB)^{t+\log(k)} \in W(L(B_{C'}))$ if and only if $(C')^t(F^k) = T^k$, if and only if $C^{s}(F^k) = T$ for some $s \leq t$. Thus we have reduced the iterated circuit problem to the intersection problem with $w^*$, proving the PSPACE-hardness of the latter.

Note that we cannot simply choose $F = \{q_1^T\}$ as the set of final states of $B_{C'}$, since then the automaton would accept any game state with an excessive set that does not contain $q_i^F$, and such sets do not correspond to an iteration of the circuit.

9 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 28. The winning set of the Dyck language is not context-free.

Proof. 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 $k \geq 2j$ is necessary. Thus the right hand side contains $L$. 

22
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] R.P. Anstee, Lajos Rónyai, and Attila Sali. Shattering News. *Graphs and Combinatorics*, 18(1):59–73, 2002.

[2] Sanjeev Arora and Boaz Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, New York, 2007.

[3] FC Auluck. On some new types of partitions associated with generalized Ferrers graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 47, pages 679–686. Cambridge University Press, 1951.

[4] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[5] Eric Goles, Pedro Montealegre, Ville Salo, and Ilkka Törmä. PSPACE-completeness of majority automata networks. *Theoretical Computer Science*, 609:118–128, 2016.

[6] Andrea Herrmann, Martin Kutrib, Andreas Malcher, and Matthias Wendlandt. Descriptive Complexity of Bounded Regular Languages. In Cezar Câmpeanu, Florin Manea, and Jeffrey Shallit, editors, *Descriptional Complexity of Formal Systems*, pages 138–152, Cham, 2016. Springer International Publishing.

[7] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[8] Friedl Katalin and Rónyai Lajos. Order shattering and Wilson’s theorem. *Discrete Mathematics*, 270(1):127–136, 2003.

[9] D. Kleitman and G. Markowsky. On Dedekind’s Problem: The Number of Isotone Boolean Functions. II. *Transactions of the American Mathematical Society*, 213:373–390, 1975.

[10] Pierre Marcus and Ilkka Törmä. Descriptive complexity of winning sets of regular languages. In Galina Jirásková and Giovanni Pighizzini, editors, *Descriptional Complexity of Formal Systems*, pages 130–141, Cham, 2020. Springer International Publishing.

[11] Ronnie Pavlov and Michael Schraudner. Classification of sofic projective subdynamics of multidimensional shifts of finite type. *Transactions of the American Mathematical Society*, 367(5):3371–3421, 2015.

[12] Jarkko Peltonäkä and Ville Salo. On winning shifts of marked uniform substitutions. *RAIRO-Theoretical Informatics and Applications*, 53(1-2):51–66, 2019.
[13] Ville Salo and Ilkka Törmä. Playing with Subshifts. *Fundamenta Informaticae*, 132(1):131–152, 2014.

[14] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1):135–183, 1998.