Recognition and Complexity Results for Projection Languages of Two-Dimensional Automata

Taylor J. Smith \textsuperscript{a} \quad Kai Salomaa \textsuperscript{a}

September 2, 2020

Abstract

The row projection (resp., column projection) of a two-dimensional language $L$ is the one-dimensional language consisting of all first rows (resp., first columns) of each two-dimensional word in $L$. The operation of row projection has previously been studied under the name “frontier language”, and previous work has focused on one- and two-dimensional language classes.

In this paper, we study projections of languages recognized by various two-dimensional automaton classes. We show that both the row and column projections of languages recognized by (four-way) two-dimensional automata are exactly context-sensitive. We also show that the column projections of languages recognized by unary three-way two-dimensional automata can be recognized using nondeterministic logspace. Finally, we study the state complexity of projection languages for two-way two-dimensional automata, focusing on the language operations of union and diagonal concatenation.

Key words and phrases: language classes, projection languages, space complexity, three-way automata, two-dimensional automata, two-way automata

MSC2020 classes: 68Q45 (primary); 68Q15, 68Q19 (secondary).

1 Introduction

A two-dimensional word is a generalization of the notion of a word from a one-dimensional string to an array or matrix of symbols. Two-dimensional words are used as the input to two-dimensional automata, whose input heads move through the input word in a variety of ways, depending on the model.

We may define special projection operations on two-dimensional words that produce either the first row or the first column of the given word. In this way, a projection can be thought of as a conversion from a two-dimensional word to a one-dimensional word. Note

\textsuperscript{a}School of Computing, Queen’s University, Kingston, Ontario, Canada. Email: \{tsmith,ksalomaa\}@cs.queensu.ca.
that projection operations are lossy (i.e., all but the first row/column of the two-dimensional word is lost when a projection operation is applied).

The row projection operation has been studied in the past [2, 13], with a particular focus on formal language theory. (We summarize previous results in Section 2.1.) However, no work has yet been done on investigating projections of languages recognized by various two-dimensional automaton models.

Our results are as follows. We show that both the row and column projections of languages recognized by (four-way) two-dimensional automata are exactly context-sensitive. We also show that the column projections of languages recognized by unary three-way two-dimensional automata belong to the class \(\text{NSPACE}(O(\log(n)))\). Finally, we study the state complexity of projection languages, focusing on the state complexity of union and diagonal concatenation for projections of languages recognized by two-way two-dimensional automata.

\section{Preliminaries}

A two-dimensional word is a matrix of symbols from some alphabet \(\Sigma\). If a two-dimensional word \(w\) has \(m\) rows and \(n\) columns, then we say that \(w\) is of dimension \(m \times n\). A two-dimensional language consists of two-dimensional words. There exist two special languages in two dimensions: \(\Sigma^{m \times n}\) consists of all words of dimension \(m \times n\) for some fixed \(m, n \geq 1\), and \(\Sigma^{**}\) consists of all two-dimensional words.

The row projection (resp., column projection) of a two-dimensional language \(L\) is the one-dimensional language consisting of the first rows (resp., first columns) of all two-dimensional words in \(L\). We formalize these definitions in terms of individual two-dimensional words. In the following pair of definitions, we assume we have an \(m \times n\) two-dimensional word

\[
\begin{bmatrix}
  a_{1,1} & \cdots & a_{1,n} \\
  \vdots & \ddots & \vdots \\
  a_{m,1} & \cdots & a_{m,n}
\end{bmatrix}
\]

**Definition 1** (Row projection). Given a two-dimensional word \(w \in \Sigma^{m \times n}\), the row projection of \(w\) is the one-dimensional word

\[
\text{pr}_R(w) = a_{1,1}a_{1,2}\cdots a_{1,n},
\]

where \(a_{1,1}, \ldots, a_{1,n} \in \Sigma\). The row projection of a two-dimensional language \(L\), denoted \(\text{pr}_R(L)\), is produced by taking the row projections of all words \(w \in L\).

**Definition 2** (Column projection). Given a two-dimensional word \(w \in \Sigma^{m \times n}\), the column projection of \(w\) is the one-dimensional word

\[
\text{pr}_C(w) = a_{1,1}a_{2,1}\cdots a_{m,1},
\]

where \(a_{1,1}, \ldots, a_{m,1} \in \Sigma\). The column projection of a two-dimensional language \(L\), denoted \(\text{pr}_C(L)\), is produced by taking the column projections of all words \(w \in L\).
Note that one may view the column projection operation as taking the “transpose” of the first column of a two-dimensional word in order to produce a one-dimensional string. The row projection operation has been considered in previous papers, where it was called the “frontier” of a word or language [13].

Two-dimensional words are used as the input to two-dimensional automata. When we provide such a word as input, we surround the outer border of the word with a special boundary symbol #. (For example, the upper-left boundary symbol is at position (0, 0) and the lower-right boundary symbol is at position (m + 1, n + 1) in the word.) The boundary symbol prevents the input head of the automaton from leaving the input word.

The formal definition of a two-dimensional automaton is as follows:

**Definition 3 (Two-dimensional automaton).** A two-dimensional automaton is a tuple \((Q, \Sigma, \delta, q_0, q_{\text{accept}})\), where \(Q\) is a finite set of states, \(\Sigma\) is the input alphabet (with \(#) \not\in \Sigma\ acting as a boundary symbol), \(\delta : (Q \setminus \{q_{\text{accept}}\}) \times (\Sigma \cup \{\#\}) \rightarrow Q \times \{U, D, L, R\}\) is the partial transition function, and \(q_0, q_{\text{accept}} \in Q\) are the initial and accepting states, respectively.

The specific model in Definition 3 is sometimes referred to as a “four-way two-dimensional automaton”. In this paper, we also consider three-way and two-way variants of two-dimensional automata. In the three-way case, the transition function is restricted to use only the directions \(\{D, L, R\}\). Likewise, in the two-way case, the transition function uses only the directions \(\{D, R\}\). We may optionally include a direction \(N\), which corresponds to “no move” and does not change the recognition power of the model. We abbreviate each automaton model as 

\(2(D/N)FA-kW(-1\Sigma)\), where \(D/N\) denotes deterministic/nondeterministic, \(k\) denotes the directions of movement, and \(1\Sigma\) denotes a unary alphabet. In later sections, we will use the notation \(L_C\) to denote the set of languages recognized by some automaton model \(C\).

### 2.1 Previous Work

A number of survey articles and other works have been written about both two-dimensional languages [5, 14] and two-dimensional automaton models [9, 15, 20]. Previous work on projection operations has taken two perspectives: language-theoretic and automata-theoretic.

**Language-theoretic.** One of the earliest results on two-dimensional row projection, due to Latteux and Simplot [13], showed that a one-dimensional language \(F\) is context-sensitive if and only if there exists a two-dimensional language \(L \in \text{REC}\) such that \(F = \text{pr}_R(L)\). The class \(\text{REC}\) denotes the class of tiling-recognizable two-dimensional languages, or languages whose words can be defined by a finite set of \(2 \times 2\) tiles [4].

Anselmo et al. [2] later extended this direction of research to give equivalent characterizations for unambiguous and deterministic context-sensitive one-dimensional languages; namely, \(F\) is unambiguous (resp., deterministic) context-sensitive if and only if there exists \(L \in \text{UREC}\) (resp., \(L \in \text{Row-UREC}_t\)) such that \(F = \text{pr}_R(L)\). The classes \(\text{UREC}\) and \(\text{Row-UREC}_t\) are subclasses of \(\text{REC}\), where \(\text{UREC}\) consists of languages defined by an unambiguous tiling system [4] and \(\text{Row-UREC}_t\) consists of languages that are “top-to-bottom row-unambiguous”; Anselmo et al. give a formal definition of the class \(\text{Row-UREC}_t\) in an earlier paper [1].
Some classes smaller than Row-UREC (namely, the class of deterministic recognizable languages DREC [1]) have no known characterization in terms of one-dimensional language classes.

**Automata-theoretic.** A (four-way) two-dimensional automaton can recognize whether or not an input word has either an exponential or a doubly-exponential side length [11]. It is well-known that the language of unary strings of exponential length is context-sensitive but not context-free [8]. This fact implies that, if $L$ is a language recognized by a four-way two-dimensional automaton, then both $\text{pr}_R(L)$ and $\text{pr}_C(L)$ may be non-context-free, even in the unary case.

Restricting ourselves to the three-way model, we obtain results that differ based on the projection operation under consideration. Let $L$ be a unary language. If $L$ is recognized by a nondeterministic three-way two-dimensional automaton, then $\text{pr}_R(L)$ is regular. On the other hand, if $L$ is recognized by a deterministic three-way two-dimensional automaton, then $\text{pr}_C(L)$ need not be regular [21]. These results apply also for general alphabets. We can improve the bound by showing that $\text{pr}_C(L)$ may be non-context-free for three-way two-dimensional automata, since the language $L_{\text{composite}}$ used in the proof of the non-regularity result is context-sensitive in both the unary and general-alphabet cases [6, 16, 17].

Finally, for the two-way model, we know that if any language $L$ is recognized by a nondeterministic two-way two-dimensional automaton, then both $\text{pr}_R(L)$ and $\text{pr}_C(L)$ are regular [21]. This applies also to deterministic and unary two-way two-dimensional automata.

### 3 Recognition Power and Space Complexity

Before we proceed further, we recall a few elementary definitions. These definitions may be found in any standard textbook on the theory of computation; e.g., Sipser [19].

Recall that a linear-bounded automaton is a nondeterministic Turing machine whose computation is restricted only to the cells of its input tape that originally contained input symbols. A configuration of a linear-bounded automaton $\mathcal{M}$ is a sequence of tape symbols of $\mathcal{M}$, where the currently-scanned symbol is distinguished by adding the current state $q$ as a subscript to the symbol. An accepting computation history of $\mathcal{M}$ on an input string $w$ is a sequence of configurations $C_0, C_1, \ldots, C_k$ that $\mathcal{M}$ enters as it performs its computation on $w$, where $C_0$ is the initial configuration of $\mathcal{M}$ on $w$, $C_{i+1}$ is obtained from $C_i$ in one computation step of $\mathcal{M}$ for all $0 \leq i \leq k - 1$, and $C_k$ is an accepting configuration. Finally, a computation table of $\mathcal{M}$ on an input word $w$ is a two-dimensional word where the rows of the word are configurations $C_0, C_1, \ldots, C_k$ appearing in the computation history of $\mathcal{M}$ on $w$.

From previous work, we know that $\text{pr}_R(L)$ is context-sensitive when $L \in \text{REC}$ [13]. It is known that $L_{\text{2DFA-4W}} \subseteq L_{\text{2NFA-4W}} \subseteq \text{REC}$ [3, 10], so $\text{pr}_R(L)$ is also context-sensitive when $L \in L_{\text{2DFA-4W}}$. The following theorem proves the other direction of this inclusion.

**Theorem 4.** Let $K$ be a context-sensitive language. Then there exists $L \in L_{\text{2DFA-4W}}$ such that $K = \text{pr}_R(L)$.

**Proof.** Let $\mathcal{M}$ be a linear-bounded automaton recognizing the language $K$. We construct a deterministic four-way two-dimensional automaton $A$ that checks whether the rows of its
input word \( w \) are configurations representing an accepting computation history of \( \mathcal{M} \). Given a two-dimensional word \( w \) over an alphabet \( \Sigma \) as input, where \( w \) is of dimension \( m \times n \), \( A \) checks each of the following properties:

1. The first row of \( w \) contains only alphabet symbols from \( \Sigma \).

2. The last row of \( w \) contains, as a subscript, an accepting state of \( \mathcal{M} \).

3. For each \( i \), where \( 1 \leq i \leq m - 1 \), the configuration \( C_i \) represented by the \((i + 1)\)st row can be obtained in one computation step of \( \mathcal{M} \) from the preceding configuration \( C_{i-1} \) represented by the \( i \)th row.

For this property, \( A \) must check the following:

(a) Any tape symbol in \( C_i \) different from the corresponding symbol in \( C_{i-1} \) was first scanned by the state in \( C_{i-1} \); and

(b) Each tape symbol in \( C_i \) different from the corresponding symbol in \( C_{i-1} \) corresponds to one valid computation step of \( \mathcal{M} \).

At the beginning of its computation, \( A \) assumes that \( \mathcal{M} \) is scanning the leftmost symbol in the first row of \( w \) from its initial state. Then, according to Property 3, \( A \) checks that the configuration represented by the second row of \( w \) can be obtained in one computation step of \( \mathcal{M} \) under this assumption.

The automaton \( A \) checks Property 3 in the following way. For each \( 1 \leq i \leq m - 1 \), the input head of \( A \) traverses the \( i \)th and \((i + 1)\)st rows in a down-and-up motion from the left boundary to the right boundary. Upon reaching a boundary, the input head returns to the input word and checks the next pair of rows. This traversal procedure is illustrated in Figure 1. Each time the input head moves downward or upward within a column, it either compares the symbols in the \( i \)th and \((i + 1)\)st rows to verify that they match or, if it is at a position where the current computation step of \( \mathcal{M} \) applies, it checks that the computation step is valid according to the transition relation of \( \mathcal{M} \).

If each of the preceding properties holds, then \( w \) encodes a computation table corresponding to an accepting computation of \( \mathcal{M} \) and \( A \) accepts \( w \). Since linear-bounded automata recognize all context-sensitive languages, and since \( \text{pr}_R(L(A)) \) is the input string to \( \mathcal{M} \), the result follows.

The proof of Theorem 4 also works for nondeterministic two-dimensional automata. Moreover, it is straightforward to show that \( \text{pr}_C(L) \) is context-sensitive when \( L \in L_{2\text{DFA-4W}} \), and so Theorem 4 can similarly be adapted to apply to column projection languages. These observations, taken together, lead to the following characterization.

**Corollary 5.** Both the row and column projections of languages recognized by four-way two-dimensional automata consist exactly of the class of context-sensitive languages.

---

\(^1\)Since the row projection of \( w \) cannot contain state information, an initial configuration of \( \mathcal{M} \), \( a_1, q_0, a_2 \ldots a_m \), is encoded on the first row as the string \( a_1a_2\ldots a_m \). The computation of \( A \) implicitly assumes that this string represents the initial configuration.
\begin{center}
\begin{tabular}{ccccccc}
\# & \# & \# & \# & \# & \# & \# \\
\# & d & a & \rightarrow & b & c & \rightarrow & a & \# \\
\# & \downarrow & \uparrow & \downarrow & \uparrow & \uparrow & \downarrow & \\
\# & b & \rightarrow & a_{q_1} & \rightarrow & b & \rightarrow & c & \rightarrow & a & \rightarrow & \# \\
\# & b & d & \rightarrow & b_{q_2} & \rightarrow & c & \rightarrow & a & \rightarrow & \# \\
\# & b & d_{q_1} & \rightarrow & c & \rightarrow & c & \rightarrow & a & \rightarrow & \# \\
\end{tabular}
\end{center}

Figure 1: An illustration of the movement of the input head of the automaton \( \mathcal{A} \), constructed in Theorem 4.

### 3.1 Three-Way Two-Dimensional Automata

Recall from Section 2.1 that the row projection of any language accepted by a three-way two-dimensional automaton \( \mathcal{A} \) is regular. Since \( \text{REG} \in \text{DSPACE}(O(1)) \) [18], we immediately get that \( \text{pr}_R(L(\mathcal{A})) \in \text{DSPACE}(O(1)) \) as well.

We further noted in the same section that the column projection of a language in \( L_{2\text{NFA}-3W-1\Sigma} \) may be non-context-free, depending on the choice of language. Here, we investigate the space complexity of column projection languages for \( L_{2\text{NFA}-3W-1\Sigma} \).

In what follows, we use the notation \( \text{rso}_i[r, s] \) to denote the subword occurrence of the \( i \)th row starting at index \( r \) and ending at index \( r + s \); that is, a subword of length \( s + 1 \). Since we are considering unary languages, all symbols of \( \text{rso}_i[r, s] \) are identical and independent of the value \( i \). Thus, by “subword occurrence”, we mean the cells of the \( i \)th row at indices \( r \) through \( r + s \) inclusive.

The following technical lemma states that every string \( w \) in the column projection of a language in \( L_{2\text{NFA}-3W-1\Sigma} \) is a projection of a two-dimensional word \( z \), where the number of columns of \( z \) is at most some constant multiple of the length of \( w \). The proof, intuitively speaking, shows that when we have a two-dimensional word containing a large number of columns with no downward moves, we can remove some of these columns and simulate the same computation of the three-way two-dimensional automaton.

**Lemma 6.** Let \( \mathcal{A} \) be a unary three-way two-dimensional automaton with \( k \) states, and consider a word \( w \in \text{pr}_C(L(\mathcal{A})) \). Then there exists a two-dimensional word \( z \) with \( |w| \) rows and at most \((|w| + 3) \cdot (k^{2k} + 2)\) columns accepted by \( \mathcal{A} \).

**Proof.** Let \( H = k^{2k} \). Consider a two-dimensional word \( z \in L(\mathcal{A}) \) of dimension \(|w| \times n_1\), where

\[
 n_1 > (|w| + 3) \cdot (H + 2). \tag{1}
\]

Let \( C_z \) be an accepting nondeterministic computation of \( \mathcal{A} \) on input word \( z \). Without loss of generality, we may assume that \( C_z \) accepts at the bottom border of \( z \).

By the inequality in Equation 1, the input word \( z \) must have \( k^{2k} + 1 \) consecutive columns such that the computation \( C_z \) does not make a downward move in any such column. Furthermore, we may assume that these consecutive columns do not include either the first \( H \)
columns or the last \( H \) columns of \( z \). That is, there exists \( H \leq j \leq (n_1 - 2H) \) such that the computation \( C_z \) does not make a downward move in any of the subword occurrences

\[ \text{rso}_i[j, H], \quad i = \{1, \ldots, |w|\}. \]

Let \( Q \) be the set of states of \( A \) and define \( \overline{Q} = \{ \overline{q} \mid q \in Q \} \) to be a disjoint copy of states in \( Q \). For each column \( x \in \{j, j + 1, \ldots, j + H\} \), define a function \( f_x : Q \to 2^{Q \times \overline{Q}} \) by setting, for all \( p \in Q \),

- \( q \in f_x(p) \) if, for some \( i \), the computation \( C_z \) on the \( i \)th row in column \( x \) and state \( p \) exits the subword occurrence \( \text{rso}_i[j, H] \) to the left in state \( q \); and

- \( \overline{q} \in f_x(p) \) if, for some \( i \), the computation \( C_z \) on the \( i \)th row in column \( x \) and state \( p \) exits the subword occurrence \( \text{rso}_i[j, H] \) to the right in state \( q \).

Note that the computation of \( C_z \) may visit the subword occurrence multiple times. By our definition, \( q \in f_x(p) \) if, at some point, \( C_z \) is in the \( x \)th column in state \( p \) and, when \( C_z \) next exits \( \text{rso}_i[j, H] \), it exits to the left in state \( q \).

Note also that the accepting computation must exit each subword occurrence \( \text{rso}_i[j, H] \) either to the left or to the right since, by our choice of \( j \), the computation \( C_z \) makes no downward moves in any of the columns \( j, \ldots, (j + H) \).

Since the number of functions from \( Q \) to \( 2^{Q \times \overline{Q}} \) is \( H = k^{2k} \), there exist columns \( x_1 \) and \( x_2 \), \( j \leq x_1 < x_2 \leq (j + H) \), such that \( f_{x_1} = f_{x_2} \). Moreover, since the computation \( C_z \) makes no downward moves in any of the columns \( j, \ldots, (j + H) \), there exists an accepting computation of \( A \) on the two-dimensional word \( z' \) obtained by removing the columns \( x_1, \ldots, (x_2 - 1) \) from \( z \).

The above observation relies on our earlier assumption that the designated columns \( j, \ldots, (j + H) \) are at distance at least \( H \) from the left and right borders of the word. For example, consider a situation where \( \overline{q} \in f_{x_2}(p) \); that is, where the computation starting in column \( x_2 \) and state \( p \) exits the subword occurrence to the right in state \( q \). When simulating the same computation on the modified word \( z' \) starting in column \( x_1 \), the computation could, at some point, move to the left of column \( j \). Since \( j \geq H \), this guarantees that the computation would not reach the left border.

Altogether, the two-dimensional word \( z' \) has \( x_2 - x_1 \) fewer columns than the original word \( z \). By repeated application of the previous argument, we see that \( A \) must accept a two-dimensional word of dimension \( |w| \times n_2 \), where \( n_2 \leq H \).

An application of Lemma 6 allows us to obtain our main space complexity result for column projections of languages recognized by unary three-way two-dimensional automata.

**Theorem 7.** Let \( A \) be a unary three-way two-dimensional automaton. Then \( \text{pr}_C(L(A)) \in \text{NSPACE}(O(\log(n))) \).

**Proof.** Suppose \( A \) has \( k \) states. We describe the operation of a nondeterministic logspace Turing machine \( M \) recognizing \( \text{pr}_C(L(A)) \).

On input word \( w \), \( M \) first writes to its work tape a binary representation of a nondeterministically-chosen natural number \( n_1 \leq (|w|+3)\cdot(k^{2k}+2) \). Since \( k \) is constant, this binary representation can be written in space \( O(\log(|w|)) \).
The machine $\mathcal{M}$ then simulates a nondeterministic computation of $\mathcal{A}$ on a two-dimensional input word $z$ with $|w|$ rows and $n_1$ columns. The input head of $\mathcal{M}$ keeps track of the current row of $z$, while a binary counter stored on the work tape of $\mathcal{M}$ keeps track of the current column of $z$. The work tape also contains the originally-guessed value $n_1$ so that $\mathcal{M}$ is able to determine when its simulated computation encounters the right border of the input word.

By Lemma 6, we know that if $w \in \text{pr}_C(L(\mathcal{A}))$, then $w$ must be a column projection of a two-dimensional word with at most $(|w| + 3) \cdot (k^{2k} + 2)$ columns that is accepted by $\mathcal{A}$. □

Since the language class CSL coincides with the space complexity class $\text{NSPACE}(O(n))$ [12], one consequence of Corollary 5 is that the row and column projections of languages recognized by four-way two-dimensional automata consist exactly of languages in $\text{NSPACE}(O(n))$. Theorem 7 gives a significantly improved space complexity upper bound for column projections of languages recognized by unary three-way two-dimensional automata.

# 4 State Complexity

Since projections of languages in $L_{\text{2DFA-2W}}$ and $L_{\text{2NFA-2W}}$ are known to be always regular, it is possible to consider questions of state complexity involving these projection languages.

Although they seem never to have appeared anywhere in the literature, it is straightforward to prove the following closure results for Boolean operations over two-way two-dimensional automata.

**Lemma 8.** The class $L_{\text{2DFA-2W}}$ is not closed under union or intersection.

**Proof.** Take $\Sigma = \{0, 1\}$. Define two languages $L_1$ and $L_2$ as follows:

$$L_1 = \{w \in \Sigma^{2 \times 2} \mid w[0, 0] = 1, w[0, 1] = 1\};$$
$$L_2 = \{w \in \Sigma^{2 \times 2} \mid w[0, 0] = 1, w[1, 0] = 1\}.$$

That is, $L_1$ is the language of $2 \times 2$ two-dimensional words with two 1s in the first two positions of its first row, and $L_2$ is the language of $2 \times 2$ two-dimensional words with two 1s in the first two positions of its first column.

Clearly, a deterministic two-way two-dimensional automaton can recognize words in $L_1$ by scanning the symbols at positions $(0, 0)$ and $(0, 1)$ and verifying that they are both 1s. The same model can recognize words from $L_2$ in a similar manner.

However, no deterministic two-way two-dimensional automaton can recognize the language $L_1 \cup L_2$. Suppose such an automaton $\mathcal{A}$ can recognize the language $L_1 \cup L_2$, and consider the computation of $\mathcal{A}$ on a $2 \times 2$ word $w \not\in L_1 \cup L_2$ where $w[0, 0] = 1$ and all other symbols are 0. This computation will reject $w$, but the input head will only scan one of the positions $(0, 1)$ or $(1, 0)$. Without loss of generality, suppose the unscanned position is $(1, 0)$. Then the automaton would also reject a $2 \times 2$ word $w' \in L_1 \cup L_2$ where $w'[0, 0] = w'[1, 0] = 1$ and all other symbols are 0.

For the same reason, no deterministic two-way two-dimensional automaton can recognize the language $L_1 \cap L_2$. Consider a $2 \times 2$ word $x \in L_1 \cap L_2$. Such a word has the symbol 1 at each of the positions $(0, 0)$, $(0, 1)$, and $(1, 0)$, but no deterministic two-way two-dimensional automaton can scan all three of these positions. Given a $2 \times 2$ word $x' \not\in L_1 \cap L_2$, where
\[ x'[0,0] = x'[0,1] = 1 \text{ and } x'[1,0] = 0, \]  
a deterministic two-way two-dimensional automaton making one rightward move cannot distinguish between \( x \) and \( x' \).

**Lemma 9.** The class \( L_{2NFA-2W} \) is closed under union, but is not closed under intersection or complement.

**Proof.** Let \( \mathcal{A} \) be a nondeterministic two-way two-dimensional automaton, and let \( L \) and \( L' \) be languages recognized by this model. Then \( \mathcal{A} \) can recognize the language \( L \cup L' \) by making a nondeterministic selection between \( L \) and \( L' \) at the beginning of its computation, and checking whether its input word \( w \) belongs to the chosen language.

Intersection is not closed for this model for the same reason as given in the proof of Lemma 8.

As a consequence of this model being closed under union but not intersection, we necessarily cannot have closure under complement.

Moreover, the present authors previously investigated closure properties of concatenation operations over two-way two-dimensional automata \([22]\). In this section, therefore, we will focus on the state complexity of projections of union and concatenation operations for nondeterministic two-way two-dimensional automata.

### 4.1 Union of 2NFA-2W Languages

Before we proceed, we require a slight modification to the definition of a two-way two-dimensional automaton that we introduced in Section 2. For the remainder of this section, when we refer to a “two-way two-dimensional automaton”, we use the following definition.

**Definition 10 (IBR-accepting two-way two-dimensional automaton).** An IBR-accepting two-way two-dimensional automaton \( \mathcal{A} \) is a tuple \( (Q, \Sigma, \delta, q_0, q_{\text{accept}}) \) as in Definition 3, where, when the input head reads a boundary marker \# for the first time, \( \mathcal{A} \) either enters \( q_{\text{accept}} \) in the next transition or the transition is undefined.

The abbreviation “IBR-accepting” refers to the automaton “immediately-bottom-right accepting”, or accepting only once the input head reaches the bottom or right border of the input word. The two-way model is the only model for which we can make this modification; neither three- nor four-way models can be made to halt immediately upon reading a boundary marker.

**Remark.** The accepting state of an IBR-accepting two-way two-dimensional automaton, \( q_{\text{accept}} \), is a “dummy” state used only as the target of accepting transitions on the boundary symbol \#. Thus, by the “size” of such an automaton \( \mathcal{A} \) we mean the size of the set \( Q - \{q_{\text{accept}}\} \). This convention ensures that an IBR-accepting two-way two-dimensional automaton recognizing single-row words has the same size as the corresponding one-dimensional automaton accepting the same string language.

The following result shows that we may convert between the usual and IBR-accepting types of two-way two-dimensional automata without incurring a penalty on the number of states.
**Proposition 11 ([22]).** Given a two-way two-dimensional automaton $A$ with $n$ states, there exists an equivalent IBR-accepting two-way two-dimensional automaton $A'$ with $n$ states.

*Proof Sketch.* If $A$ reads a boundary marker, then its input head can never reenter the input word. After reading a boundary marker in state $q_i$, say, we can decide whether $q_{\text{accept}}$ is reachable from $q_i$ after following some number of transitions of $A$. Thus, we may take $A'$ to be the same as $A$ where the transition upon reading the boundary marker $#$ goes directly to $q_{\text{accept}}$ if that state is reachable or is undefined otherwise. 

Using a construction from a previous paper investigating projections of nondeterministic two-way two-dimensional automaton languages [21], we may obtain an upper bound on the nondeterministic state complexity of projection languages for this model.

**Proposition 12.** Let $A$ be a nondeterministic two-way two-dimensional automaton with $n$ states. Then both $\text{pr}_R(L(A))$ and $\text{pr}_C(L(A))$ are recognized by a nondeterministic one-dimensional automaton with $2n$ states.

*Proof Sketch.* Given a nondeterministic two-way two-dimensional automaton $A$, we may construct a nondeterministic one-dimensional automaton $B$ recognizing the language $\text{pr}_R(L(A))$ that simulates rightward moves of $A$ and keeps track of whether a downward move is made during the computation of $A$. We may remember downward moves by doubling the number of states of $A$. Using an analogous construction, we obtain the same result for $\text{pr}_C(L(A))$. 

We can show that the following lower bound applies for the same model.

**Lemma 13.** There exists a nondeterministic two-way two-dimensional automaton $A$ with $n$ states such that any nondeterministic one-dimensional automaton recognizing $\text{pr}_R(L(A))$ requires at least $2n - 1$ states.

*Proof.* Define $A$ as follows: the alphabet is $\Sigma = \{0, 1\}$, the set of states is $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ (and additionally $q_{\text{accept}}$), the initial state is $q_0$, the accepting state is $q_{\text{accept}}$, and the transition function $\delta$ consists of the following:

- $\delta(q_i, 0) = (q_{i+1}, R)$ for all $0 \leq i < n - 2$;
- $\delta(q_{n-1}, 0) = \{(q_0, R), (q_{n-1}, D)\}$; and
- $\delta(q_0, \#) = (q_{\text{accept}}, N)$.

Each rightward-moving transition counts modulo $n$, and the only downward-moving transition occurs in a column position congruent to $-1 \mod n$. Moreover, the downward-moving transition does not change the state (i.e., the column count is preserved). Note also that $A$ makes no transitions upon reading the symbol 1; this is because, after reading $n - 1$ copies of 0 and making a downward move, the first row can contain any symbols after that column position so long as the number of total columns remains a multiple of $n$. Combining these observations, we see that the row projection of $L(A)$ is

$$L_{\text{pr}} = 0^{n-1}(0+1)((0+1)^n)^* + \epsilon.$$
To show that the nondeterministic state complexity of $L_{pr}$ is at least $2n - 1$, we use the following extended fooling set [7]:

$$S = \{(x, y) \mid xy = 0^{n-1}1^{n+1}, |y| \geq 2\}.$$ 

The set $S$ contains $2n - 1$ elements and, by its definition, for any pair $(x, y) \in S$, $xy = 0^{n-1}1^{n+1} \in L_{pr}$.

Consider two distinct pairs $(x, y)$ and $(x', y')$. Without loss of generality, assume $x$ is a proper prefix of $x'$. If $|x'| - |x| \neq n$, then $|xy'|$ is not a multiple of $n$, and $xy' \notin L_{pr}$. Otherwise, $|x'| - |x| = n$. In this case, since $|x'y'| = 2n$ and $|y'| \geq 2$, we have that $|x'| \leq 2n - 2$. Thus, $|x| \leq n - 2$, and so $x = 0^i$ for some $0 \leq i \leq n - 2$. However, this means that $xy' \notin L_{pr}$, because in this case $y'$ consists only of the symbol 1. □

Using the previous results, we can obtain a state complexity bound for the projection of the union of two languages recognized by nondeterministic two-way two-dimensional automata.

**Theorem 14.** (i) If $A$ and $B$ are nondeterministic two-way two-dimensional automata with $m$ and $n$ states, respectively, then $pr_R(L(A) \cup L(B))$ is recognized by a nondeterministic one-dimensional automaton with $2(m + n + 1)$ states.

(ii) There exist nondeterministic two-way two-dimensional automata $A$ and $B$ with $n$ and $m$ states, respectively, such that any nondeterministic one-dimensional automaton recognizing $pr_R(L(A) \cup L(B))$ requires at least $2(m + n - 1)$ states.

**Proof.** We prove (i) by construction. Without loss of generality, assume the state sets of $A$ and $B$ are disjoint. Then a nondeterministic two-way two-dimensional automaton $C$ recognizing the language $L(A) \cup L(B)$ can be constructed in the following way:

- The state set of $C$ contains all non-accepting states of $A$ and $B$.
- There exists a new initial state $q_0_c$ that simulates outgoing transitions from the original initial states $q_{0A}$ and $q_{0B}$.
- There exists a joint accepting state $q_{accept_c}$.
- The transition function $\delta_C$ includes all transitions of $\delta_A$ and $\delta_B$.

Since the state sets of $A$ and $B$ are disjoint, $C$ accepts some input $w$ if and only if $w$ is accepted by either $A$ or $B$. By our construction, $C$ consists of $m + n + 1$ states. Since $C$ must remember whether or not a downward move is made during the computation of $A$ (or, similarly, during the computation of $B$), we must double the number of states of $C$; this is essentially the same construction as that used by Proposition 12. Therefore, $pr_R(L(A) \cup L(B))$ is recognized by a nondeterministic one-dimensional automaton with $2(m + n + 1)$ states.

We now prove (ii). Let $A$ be the $n$-state nondeterministic two-way two-dimensional automaton from the proof of Lemma 13, and let $B$ be a “copy” of $A$ with $m$ states; specifically, $B$ is an automaton of the same type as $A$ over the alphabet $\Sigma' = \{2, 3\}$ where the set of states is $Q' = \{q_0, q_1, \ldots, q_{m-1}\}$ (and additionally $q_{accept}$); the transition function $\delta'$ is identical to $\delta$ with 0, 1, and $n$ replaced by 2, 3, and $m$, respectively; and all other aspects are the same.
Let \( L_{pr} = \text{pr}_R(L(A) \cup L(B)) \). A fooling set for \( L_{pr} \) is

\[
S = \{ (x, y) \mid xy = 0^{n-1}1^{n+1}, |y| \geq 2 \} \cup \{ (x, y) \mid xy = 2^{m-1}3^{m+1}, |y| \geq 2 \}.
\]

The set \( S \) contains \( 2(m + n - 1) \) elements. Moreover, \( S \) is clearly a fooling set, since mixing a pair over the alphabet \( \{0, 1\} \) and a pair over the alphabet \( \{2, 3\} \) always produces strings not in \( L_{pr} \).

Since two-way two-dimensional automata operate symmetrically with respect to rows and columns, there also exist nondeterministic state complexity bounds for column projections analogous to those established in Theorem 14.

### 4.2 Diagonal Concatenation of 2NFA-2W Languages

Given two-dimensional words \( w \) and \( v \) of dimension \( m \times n \) and \( m' \times n' \) respectively, the diagonal concatenation of \( w \) and \( v \), denoted \( w \otimes v \), produces a two-dimensional language consisting of words of dimension \( (m + m') \times (n + n') \) where \( w \) is in the top-left corner, \( v \) is in the bottom-right corner, and words \( x \in \Sigma^{m \times n} \) and \( y \in \Sigma^{m' \times n'} \) are placed in the “top-right” and “bottom-left” corners of \( w \otimes v \), respectively. We assume that the symbols in \( x \) and \( y \) come from the same alphabet \( \Sigma \) as the symbols in \( w \) and \( v \). The diagonal concatenation language is formed by adding to the corners all possible words \( x \) and \( y \) over \( \Sigma \). An example word from such a language is depicted in Figure 2.

Nondeterministic two-way two-dimensional automata are known to be closed under diagonal concatenation over a general alphabet and, moreover, this is the only concatenation operation under which two-way two-dimensional automaton languages over general alphabets are closed \[22\]. Thus, the natural question arises: given a pair of nondeterministic two-way two-dimensional automata \( A \) and \( B \) recognizing languages \( L(A) \) and \( L(B) \), respectively, how large must such an automaton be to recognize \( \text{pr}_R(L(A) \otimes L(B)) \)?

We begin by making an elementary observation. In one dimension, an \( \epsilon \)-NFA extends an ordinary NFA by allowing \( \epsilon \)-transitions; i.e., “stay-in-place” moves. The following result is well-known:

**Lemma 15** (Wood \[23\]). Any \( n \)-state \( \epsilon \)-NFA has an equivalent \( n \)-state NFA without \( \epsilon \)-transitions.
Moreover, for a pair of nondeterministic one-dimensional automata with \( m' \) and \( n' \) states recognizing languages \( L_1 \) and \( L_2 \), respectively, a total of \( m' + n' \) states are necessary and sufficient to recognize the concatenation language \( L_1 \cdot L_2 \) in the general alphabet case, while \( m' + n' - 1 \) states are necessary in the unary case [7].

**Theorem 16.** (i) If \( A \) and \( B \) are nondeterministic two-way two-dimensional automata with \( m \) and \( n \) states, respectively, then \( \text{pr}_R(L(A) \odot L(B)) \) is recognized by a nondeterministic one-dimensional automaton with \( 2m + n \) states.

(ii) There exist nondeterministic two-way two-dimensional automata \( A \) and \( B \) with \( m \) and \( n \) states, respectively, such that any nondeterministic one-dimensional automaton recognizing \( \text{pr}_R(L(A) \odot L(B)) \) requires at least \( m + n - 1 \) states.

**Proof.** We prove (i) by constructing a nondeterministic one-dimensional automaton \( C \) to recognize the language \( \text{pr}_R(L(A) \odot L(B)) \). The following procedure allows \( C \) to simulate the computation of \( A \) and \( B \) on a word in the language \( L(A) \odot L(B) \):

1. The input head of \( C \) begins by simulating rightward moves of the input head of \( A \). If the input head of \( A \) makes a downward move, \( C \) remembers that a downward move occurred and replaces it with a “stay-in-place” move.

2. At some point during its computation, \( C \) nondeterministically switches to simulating moves of \( B \). Again, the input head of \( C \) only simulates rightward moves, and replaces downward moves with “stay-in-place” moves.

By Lemma 15, “stay-in-place” moves can be used without affecting the number of states. However, by a construction similar to that used in Proposition 12, the requirement in Step 1 to remember whether a downward move occurred doubles the number of states needed to simulate the computation of \( A \). Remembering downward moves is not required when simulating the computation of \( B \). Furthermore, in Step 2, the input head of \( C \) ignores the alphabet symbols it is reading. Since the simulation only needs to check that \( B \) accepts a two-dimensional word with the correct number of columns, the exact symbols being read at this stage may be ignored.

If the computation of \( C \) accepts, then the computation of \( A \) and \( B \) must have also accepted, and therefore \( C \) recognizes words in the language \( \text{pr}_R(L(A) \odot L(B)) \). Moreover, \( 2m + n \) states are sufficient for \( C \) to perform its computation in this way.

We now prove (ii). Let \( A' \) (respectively, \( B' \)) be an \( m \)-state (respectively, \( n \)-state) unary nondeterministic one-dimensional automaton such that the concatenation of \( L(A') \) and \( L(B') \) requires \( m + n - 1 \) states [7]. The language \( L(A') \) can be recognized by an \( m \)-state nondeterministic two-way two-dimensional automaton \( A \) that recognizes words consisting of one row. Similarly, \( L(B') \) can be recognized by an \( n \)-state nondeterministic two-way two-dimensional automaton \( B \). In this case, the languages \( \text{pr}_R(L(A) \odot L(B)) \) and \( L(A') \cdot L(B') \) are equal. It follows that \( m + n - 1 \) states are necessary for any nondeterministic one-dimensional automaton to recognize \( \text{pr}_R(L(A) \odot L(B)) \).

Again, there exist nondeterministic state complexity bounds for column projections analogous to those established in Theorem 16.
5 Conclusion

In this paper, we established results linking one-dimensional language classes to two-dimensional projection languages; namely, that both the row and column projections of languages $L \in L_{2\text{DFA}-4W}$ or $L_{2\text{NFA}-4W}$ are exactly context-sensitive. This improves on the previously-known non-context-free lower bound, which remains for other two-dimensional automaton models.

We also proved space complexity results for projection languages. While both the row and column projections of languages $L \in L_{2\text{DFA}-4W}$ or $L_{2\text{NFA}-4W}$ belong to the class $\mathsf{NSPACE}(O(n))$, the column projection of languages $L \in L_{2\text{DFA}-3W-1\Sigma}$ or $L_{2\text{NFA}-3W-1\Sigma}$ belongs to the class $\mathsf{NSPACE}(O(\log(n)))$.

Finally, we investigated the state complexity of projection languages. We showed that, given a pair of nondeterministic two-way two-dimensional automata $A$ and $B$ with $m$ and $n$ states, respectively, between $2(m + n - 1)$ and $2(m + n + 1)$ states are needed to recognize $\text{pr}_R(L(A) \cup L(B))$ and between $m + n - 1$ and $2m + n$ states are needed to recognize $\text{pr}_R(L(A) \ominus L(B))$. These bounds apply also to the column projections of such languages.

We conclude by giving a selection of open problems arising from work done in this paper.

1. Which class of one-dimensional languages corresponds to $L_{2\text{DFA}-3W}/L_{2\text{NFA}-3W}$ (or their unary equivalents) under the operation of column projection?

2. Which class of one-dimensional languages corresponds to $L_{2\text{DFA}-4W-1\Sigma}/L_{2\text{NFA}-4W-1\Sigma}$ under the operations of row and column projection?

3. If a two-dimensional automaton $A$ with $n$ states recognizes a language $L$, how many states are necessary/sufficient for a one-dimensional automaton $A'$ to recognize the language $\text{pr}_R(L)/\text{pr}_C(L)$?

Problems 1 and 2 are likely difficult; it may be more reasonable to obtain an improved upper bound on the related question of space complexity for problem 2, say $\mathsf{DSPACE}(O(n))$. Moreover, for problem 3, we can obtain a trivial lower bound of $n$ states by constructing an $n$-state nondeterministic three/four-way two-dimensional automaton $A$ that accepts only words of dimension $1 \times k, k \geq 1$, and taking $A'$ to be the minimal nondeterministic two-way one-dimensional automaton recognizing the language $\text{pr}_R(L(A))$.

References

[1] Marcella Anselmo, Dora Giammarresi, and Maria Madonia. Deterministic and unambiguous families within recognizable two-dimensional languages. *Fundamenta Informatica*, 98(2–3):143–166, 2010.

[2] Marcella Anselmo, Dora Giammarresi, and Maria Madonia. Classification of string languages via tiling recognizable picture languages. In A.-H. Dediu, S. Inenaga, and C. Martín-Vide, editors, *Proceedings of the 5th International Conference on Language and Automata Theory and Applications (LATA 2011)*, volume 6638 of *Lecture Notes in Computer Science*, pages 105–116, Berlin Heidelberg, 2011. Springer-Verlag.
[3] Manuel Blum and Carl Hewitt. Automata on a 2-dimensional tape. In R. E. Miller, editor, Proceedings of the 8th Annual Symposium on Switching and Automata Theory (SWAT 1967), pages 155–160, 1967.

[4] Dora Giammarresi and Antonio Restivo. Recognizable picture languages. International Journal of Pattern Recognition and Artificial Intelligence, 6(2–3):241–256, 1992.

[5] Dora Giammarresi and Antonio Restivo. Two-dimensional languages. In G. Rozenberg and A. Salomaa, editors, Handbook of Formal Languages, volume 3, pages 215–267. Springer-Verlag, Berlin Heidelberg, 1997.

[6] Juris Hartmanis and Herbert Shank. On the recognition of primes by automata. Journal of the ACM, 15(3):382–389, 1968.

[7] Markus Holzer and Martin Kutrib. Nondeterministic descriptional complexity of regular languages. International Journal of Foundations of Computer Science, 14(6):1087–1102, 2003.

[8] John E. Hopcroft and Jeffrey D. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, 1979.

[9] Katsushi Inoue and Itsuo Takanami. A survey of two-dimensional automata theory. Information Sciences, 55(1–3):99–121, 1991.

[10] Katsushi Inoue and Itsuo Takanami. A characterization of recognizable picture languages. In A. Nakamura, M. Nivat, M. Saoudi, P. S. P. Wang, and K. Inoue, editors, Proceedings of the 2nd International Conference on Parallel Image Analysis (ICPIA 1992), volume 654 of Lecture Notes in Computer Science, pages 133–143, Berlin Heidelberg, 1992. Springer-Verlag.

[11] Jarkko Kari and Cristopher Moore. Rectangles and squares recognized by two-dimensional automata. In J. Karhumäki, H. Maurer, G. Paun, and G. Rozenberg, editors, Theory is Forever: Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday, volume 3113 of Lecture Notes in Computer Science, pages 134–144, Berlin Heidelberg, 2004. Springer-Verlag.

[12] Sige-Yuki Kuroda. Classes of languages and linear-bounded automata. Information and Control, 7(2):207–223, 1965.

[13] Michel Latteux and David Simplot. Context-sensitive string languages and recognizable picture languages. Information and Computation, 138(2):160–169, 1997.

[14] Kenichi Morita. Two-dimensional languages. In C. Martín-Vide, V. Mitran, and G. Päun, editors, Formal Languages and Applications, volume 148 of Studies in Fuzziness and Soft Computing, pages 427–437. Springer-Verlag, Berlin Heidelberg, 2004.

[15] Azriel Rosenfeld. Picture Languages: Formal Models for Picture Recognition. Computer Science and Applied Mathematics. Academic Press, New York, 1979.
[16] Arto Salomaa. *Theory of Automata*, volume 100 of *International Series of Monographs in Pure and Applied Mathematics*. Pergamon Press, Oxford, 1969.

[17] Arto Salomaa. *Formal Languages*. Academic Press, New York, 1973.

[18] John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

[19] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, Boston, 1997.

[20] Taylor J. Smith. Two-dimensional automata. Technical report 2019-637, Queen’s University, Kingston, 2019.

[21] Taylor J. Smith and Kai Salomaa. Decision problems for restricted variants of two-dimensional automata. In M. Hospodár and G. Jirásková, editors, *Proceedings of the 24th International Conference on Implementation and Application of Automata (CIAA 2019)*, volume 11601 of *Lecture Notes in Computer Science*, pages 222–234, Berlin Heidelberg, 2019. Springer-Verlag.

[22] Taylor J. Smith and Kai Salomaa. Concatenation operations and restricted variants of two-dimensional automata, 2020. arXiv:2008.11164.

[23] Derick Wood. *Theory of Computation*. Harper & Row Computer Science and Technology Series. Harper & Row, New York, 1987.