Formal Languages and Automata Theory in Two Dimensions

StFX Department of Computer Science Seminar

Taylor J. Smith

Joint work with many people

Department of Computer Science
St. Francis Xavier University
Antigonish, Nova Scotia, Canada

February 21, 2023

Table of Contents

My Research Work

Background
- Two-Dimensional Automata
- Restricted 2D Automata

Past Results and Future Questions
- Decision Problems
- Concatenation of 2D Languages
- Projections of 2D Languages
- State Complexity of 2D Automata

Other Research Problems
- Combinatorics on Words
- Bio-inspired Language Operations
- Symbolic Computation Using Automata

Table of Contents

My Research Work

Background
- Two-Dimensional Automata
- Restricted 2D Automata

Past Results and Future Questions
- Decision Problems
- Concatenation of 2D Languages
- Projections of 2D Languages
- State Complexity of 2D Automata

Other Research Problems
- Combinatorics on Words
- Bio-inspired Language Operations
- Symbolic Computation Using Automata

What is Theory?

- Theoretical computer science is all about the mathematical aspects underpinning the study and use of computers.
- "Theory" is a broad umbrella term encompassing:
  - algorithm analysis
  - algorithm design
  - automata theory
  - complexity theory
  - computability theory
  - data structures
  - formal language theory
  - information theory
  - programming language design
  - . . . and even more!

What is Theory?

- Theoretical computer science is all about the mathematical aspects underpinning the study and use of computers.
- "Theory" is a broad umbrella term encompassing:
  - algorithm analysis
  - algorithm design
  - **automata theory**
  - complexity theory
  - computability theory
  - data structures
  - **formal language theory**
  - information theory
  - programming language design
  - . . . and even more!
- **My work** focuses on formal languages and automata theory.

What is Theory?

- Theoretical computer science is all about the mathematical aspects underpinning the study and use of computers.
- "Theory" is a broad umbrella term encompassing:
  - *algorithm analysis*
  - *algorithm design*
  - **automata theory**
  - *complexity theory*
  - computability theory
  - data structures
  - **formal language theory**
  - information theory
  - programming language design
  - . . . and even more!
- **My work** focuses on formal languages and automata theory.
- I've also worked with *some other aspects* of theory.

My Research

- My research is primarily in **automata theory**.
  - Specifically, **two-dimensional** automata theory.
- Automata theory studies abstract computing machines and what we can do with/on them.

My Research

- My research is primarily in **automata theory**.
  - Specifically, **two-dimensional** automata theory.
- Automata theory studies abstract computing machines and what we can do with/on them.
- I am also interested in **formal languages** and **combinatorics on words**.
- Formal language theory studies the syntax, semantics, and expressiveness of the languages (or sets) abstract computing machines recognize.
- Combinatorics on words applies combinatorial techniques to these same languages to study their properties.

Table of Contents

My Research Work

Background
- Two-Dimensional Automata
- Restricted 2D Automata

Past Results and Future Questions
- Decision Problems
- Concatenation of 2D Languages
- Projections of 2D Languages
- State Complexity of 2D Automata

Other Research Problems
- Combinatorics on Words
- Bio-inspired Language Operations
- Symbolic Computation Using Automata

Two-Dimensional Automata

- A **two-dimensional** (2D) **automaton** is a generalization of a one-dimensional automaton.
- Two major differences:
  1. Different input word
  2. Different transition function

Two-Dimensional Automata

- A **two-dimensional** (2D) **automaton** is a generalization of a one-dimensional automaton.
- Two major differences:
  1. **Different input word**
  2. Different transition function

$$\begin{array}{cccccc}
\# & \# & \# & \cdots & \# & \# \\
\# & a_{1,1} & a_{1,2} & \cdots & a_{1,n} & \# \\
\# & a_{2,1} & a_{2,2} & \cdots & a_{2,n} & \# \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\# & a_{m,1} & a_{m,2} & \cdots & a_{m,n} & \# \\
\# & \# & \# & \cdots & \# & \#
\end{array}$$

Two-Dimensional Automata

- A **two-dimensional** (2D) **automaton** is a generalization of a one-dimensional automaton.
- Two major differences:
  1. Different input word
  2. **Different transition function**

$$\delta : (Q \setminus q_{\text{accept}}) \times (\Sigma \cup \{\#\}) \to Q \times \{U, D, L, R\}$$

$$\delta : (Q \setminus q_{\text{accept}}) \times (\Sigma \cup \{\#\}) \to 2^{Q \times \{U,D,L,R\}}$$

Deterministic four-way (2DFA-4W)

Nondeterministic four-way (2NFA-4W)

Restricted 2D Automata

- 2D automata do not have to be **four-way automata**.
- Restrict the transition function to get:
  - **Three-way (3W) automata**: $\{D, L, R\}$
  - **Two-way (2W) automata**: $\{D, R\}$
- Three-way automata cannot return to a row after moving downward, but they can read symbols multiple times in a row.
- Two-way automata are "read-once".

Restricted 2D Automata

- 2D automata do not have to be **four-way automata**.
- Restrict the transition function to get:
  - **Three-way (3W) automata**: $\{D, L, R\}$
  - **Two-way (2W) automata**: $\{D, R\}$
- Three-way automata cannot return to a row after moving downward, but they can read symbols multiple times in a row.
- Two-way automata are "read-once".

Research question: What other variant models can we study?

Restricted 2D Automata

- 2D automata do not have to be **four-way automata**.
- Restrict the transition function to get:
  - **Three-way (3W) automata**: $\{D, L, R\}$
  - **Two-way (2W) automata**: $\{D, R\}$
- Three-way automata cannot return to a row after moving downward, but they can read symbols multiple times in a row.
- Two-way automata are "read-once".

Research question: What other variant models can we study?

Recent research: Two-dimensional typewriter automata variant.

(Smith, 2022)

Table of Contents

StFX UNIVERSITY

My Research Work

Background
- Two-Dimensional Automata
- Restricted 2D Automata

Past Results and Future Questions
- Decision Problems
- Concatenation of 2D Languages
- Projections of 2D Languages
- State Complexity of 2D Automata

Other Research Problems
- Combinatorics on Words
- Bio-inspired Language Operations
- Symbolic Computation Using Automata

Decision Problems

- An automaton $\mathcal{A}$ recognizes a language $L(\mathcal{A})$.
- **Decision problems** model questions we ask about languages.
- If a problem is **decidable**, then there exists an algorithmic procedure to solve that problem.
- Some common decision problems for two languages $L(\mathcal{A})$ and $L(\mathcal{B})$:
  - **Membership:** $w \in L(\mathcal{A})$ for some 2D word w
  - **Emptiness:** $L(\mathcal{A}) = \emptyset$
  - **Universality:** $L(\mathcal{A}) = \Sigma^{**}$ (the set of all 2D words)
  - **Equivalence:** $L(\mathcal{A}) = L(\mathcal{B})$
  - **Inclusion:** $L(\mathcal{A}) \subseteq L(\mathcal{B})$
  - **Disjointness:** $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

Decision Problems: Decidability

	2DFA-4W	2NFA-4W	2DFA-3W	2NFA-3W	2DFA-2W	2NFA-2W
membership	✓	✓	✓	✓	✓	✓
emptiness	✗	✗	✓	?	?	?
universality	✗	✗	✓	✗	✓	?
equivalence	✗	✗	?	✗	?	?
inclusion	✗	✗	✗	✗	?	?
disjointness	✗	✗	✗	✗	?	?

Decision Problems: Decidability

	2DFA-4W	2NFA-4W	2DFA-3W	2NFA-3W	2DFA-2W	2NFA-2W
membership	✓	✓	✓	✓	✓	✓
emptiness	✗	✗	✓	✓	✓	✓
universality	✗	✗	✓	✗	✓	✗
equivalence	✗	✗	?	✗	✓	✗
inclusion	✗	✗	✗	✗	✓	✗
disjointness	✗	✗	✗	✗	✓	?

(Smith and Salomaa, *TCS* 2021)

Decision Problems: Decidability

	2DFA-4W	2NFA-4W	2DFA-3W	2NFA-3W	2DFA-2W	2NFA-2W
membership	✓	✓	✓	✓	✓	✓
emptiness	✗	✗	✓	✓	✓	✓
universality	✗	✗	✓	✗	✓	✗
equivalence	✗	✗	?	✗	✓	✗
inclusion	✗	✗	✗	✗	✓	✗
disjointness	✗	✗	✗	✗	✓	?

Research question: Are the question marks ✓ or ✗?

(Smith and Salomaa, *TCS* 2021)

Operations on Languages

- There are a number of operations we can apply to 2D languages.
- Some of these operations are basic set operations:
  - **Union:** $L_1 \cup L_2$ contains all words in *either* language
  - **Intersection:** $L_1 \cap L_2$ contains all words in *both* languages
  - **Complement:** $\overline{L}$ contains all words *not* in L
- Other operations are unique to formal language theory:
  - **Concatenation:** $L_1 \circ L_2$ places all words in L_1 *adjacent to* all words in L_2 in some way
  - **Reversal:** L^{R} reverses the order of the rows in all words of L
  - **Rotation:** $L^{\circlearrowright}$ rotates all words in L by 90° clockwise
- An operation is **closed** for an automaton model if the model recognizes both the original language(s) and the operator language.

Concatenation Operations

- Let's focus on "the" concatenation operation $L_1 \circ L_2$.
- We can concatenate 2D words in two different ways: row-wise or column-wise.

Concatenation Operations

- Let's focus on "the" concatenation operation $L_1 \circ L_2$.
- We can concatenate 2D words in two different ways: **row-wise** or column-wise.

$$w \ominus v = \begin{matrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \\ v_{1,1} & \cdots & v_{1,n} \\ \vdots & \ddots & \vdots \\ v_{m',1} & \cdots & v_{m',n} \end{matrix}$$

Concatenation Operations

- Let's focus on "the" concatenation operation $L_1 \circ L_2$.
- We can concatenate 2D words in two different ways: row-wise or **column-wise**.

$$w \obar v = \begin{matrix} w_{1,1} & \cdots & w_{1,n} & v_{1,1} & \cdots & v_{1,n'} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} & v_{m,1} & \cdots & v_{m,n'} \end{matrix}$$

Concatenation Operations

STFX UNIVERSITY

- Let's focus on "the" concatenation operation $L_1 \circ L_2$.
- We can also concatenate two 2D words **diagonally**.

$$w \oslash v = \begin{matrix} w_{1,1} & \cdots & w_{1,n} & x_{1,1} & \cdots & x_{1,n'} \\ \vdots & & \vdots & \vdots & & \vdots \\ w_{m,1} & \cdots & w_{m,n} & x_{m,1} & \cdots & x_{m,n'} \\ y_{1,1} & \cdots & y_{1,n} & v_{1,1} & \cdots & v_{1,n'} \\ \vdots & & \vdots & \vdots & & \vdots \\ y_{m',1} & \cdots & y_{m',n} & v_{m',1} & \cdots & v_{m',n'} \end{matrix}$$

Concatenation Operations: Closure

	2DFA-4W	2NFA-4W	2DFA-3W	2NFA-3W	2DFA-2W	2NFA-2W
Row ($\ominus$)	✗	✗	✗	✓	?	?
Column ($\varobar$)	✗	✗	✗	✗	?	?
Diagonal ($\oslash$)	?	?	?	?	?	?

Concatenation Operations: Closure

	2DFA-4W	2NFA-4W	2DFA-3W	2NFA-3W	2DFA-2W	2NFA-2W
Row (⊖)	✗	✗	✗	✓	✗	✗ / ✓ unary
Column (⦶)	✗	✗	✗	✗	✗	✗ / ✓ unary
Diagonal (⊘)	?	?	✗	?	✗	✓

(Smith and Salomaa, SOFSEM 2021)

Concatenation Operations: Closure

StFX UNIVERSITY

	2DFA-4W	2NFA-4W	2DFA-3W	2NFA-3W	2DFA-2W	2NFA-2W
Row (⊖)	✗	✗	✗	✓	(✗)	(✗) / (✓ unary)
Column (⦶)	✗	✗	✗	✗	(✗)	(✗) / (✓ unary)
Diagonal (⊘)	?	?	(✗)	?	(✗)	(✓)

Research question: Are the question marks ✓ or ✗?

(Smith and Salomaa, SOFSEM 2021)

Projection Operations

- We can define special **projection operations** on 2D words that produce the first row or the first column of the word.
- The row/column projection of a 2D language L is the 1D language consisting of all first rows/columns of all 2D words in L.

$$w = \begin{matrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{matrix}$$

$$\mathrm{pr}_{\mathrm{R}}(w) = w_{1,1} w_{1,2} \cdots w_{1,n}$$
$$\mathrm{pr}_{\mathrm{C}}(w) = w_{1,1} w_{2,1} \cdots w_{m,1}$$

Projection Operations: Space Complexity

	$\mathcal{A}$	$\mathrm{pr}_R(L(\mathcal{A}))$	$\mathrm{pr}_C(L(\mathcal{A}))$
General	-4W	NSPACE($O(n)$)	NSPACE($O(n)$)
	-3W	DSPACE($O(1)$)	**?**
	-2W	DSPACE($O(1)$)	DSPACE($O(1)$)
Unary	-4W	**?**	**?**
	-3W	DSPACE($O(1)$)	$\leq$ NSPACE($O(\log(n))$)
	-2W	DSPACE($O(1)$)	DSPACE($O(1)$)

- The **regular languages** are in DSPACE($O(1)$).
- The **context-sensitive languages** are in NSPACE($O(n)$).

(Smith and Salomaa, 2020)

Projection Operations: Space Complexity

	$\mathcal{A}$	$\mathrm{pr}_{\mathrm{R}}(L(\mathcal{A}))$	$\mathrm{pr}_{\mathrm{C}}(L(\mathcal{A}))$
General	-4W	NSPACE($O(n)$)	NSPACE($O(n)$)
	-3W	DSPACE($O(1)$)	**?**
	-2W	DSPACE($O(1)$)	DSPACE($O(1)$)
Unary	-4W	**?**	**?**
	-3W	DSPACE($O(1)$)	$\leq$ NSPACE($O(\log(n))$)
	-2W	DSPACE($O(1)$)	DSPACE($O(1)$)

Research question: What is the space complexity for the question mark entries?

(Smith and Salomaa, 2020)

State Complexity

- **State complexity** is a measure of computational complexity, much like time or space complexity.
- It is a measure specific to automata.
- There are two “types” of state complexity:
  - The **state complexity tradeoff** between two models asks for the least number of states in some automaton model sufficient to recognize all languages recognized by an n-state automaton model of another type.
  - The **operational state complexity** of a closed language operation $\circ$ asks, for an m-state automaton $\mathcal{A}$ and an n-state automaton $\mathcal{B}$, how many states are necessary/sufficient to recognize the language $L(\mathcal{A}) \circ L(\mathcal{B})$.

(Salomaa, Salomaa, and Smith, 2023)

State Complexity: Examples

- State complexity tradeoff:
  - An n-state NFA has an equivalent DFA with at most 2^n states. (Rabin and Scott, 1959)
- Operational state complexity:
  - For DFAs $\mathcal{A}$ and $\mathcal{B}$:
    - $L(\mathcal{A}) \cup L(\mathcal{B})$ requires mn states.
    - $L(\mathcal{A}) \cap L(\mathcal{B})$ requires mn states. (Maslov, 1970)
    - $\overline{L(\mathcal{A})}$ requires m states. (folklore)
  - For NFAs $\mathcal{A}$ and $\mathcal{B}$:
    - $L(\mathcal{A}) \cup L(\mathcal{B})$ requires $m + n + 1$ states.
    - $L(\mathcal{A}) \cap L(\mathcal{B})$ requires mn states. (Holzer and Kutrib, 2003)
    - $\overline{L(\mathcal{A})}$ requires 2^m states. (Birget, 1993)

State Complexity: Two Dimensions

- State complexity is very well-studied in one dimension.
  - Natural measure for automata and regular languages.
- In two dimensions. . . what do we do?
  - 2D automata are much more powerful than 1D automata!
  - We can't use the same techniques directly.

State Complexity: Two Dimensions

- State complexity is very well-studied in one dimension.
  - Natural measure for automata and regular languages.
- In two dimensions. . . what do we do?
  - 2D automata are much more powerful than 1D automata!
  - We can't use the same techniques directly.
- **Idea:** Use projection languages!
  - Row projections of three-/two-way 2D languages are regular.
  - Column projections of two-way 2D languages are regular.

State Complexity: Two Dimensions

- State complexity tradeoff:
  - n-state two-way 2D automaton $\rightarrow$ NFA:
    between $2n - 1$ and $2n$ states
- Operational state complexity:
  - $\mathrm{pr}_{\mathrm{R}}(L(\mathcal{A}) \cup L(\mathcal{B}))$ for two-way 2D automata:
    between $2(m + n - 1)$ and $2(m + n + 1)$ states
  - $\mathrm{pr}_{\mathrm{R}}(L(\mathcal{A}) \oslash L(\mathcal{B}))$ for two-way 2D automata:
    between $m + n - 1$ and $2m + n$ states

(Smith and Salomaa, *JALC* article to appear)

State Complexity: Two Dimensions

- State complexity tradeoff:
  - n-state two-way 2D automaton $\rightarrow$ NFA:
    between $2n-1$ and $2n$ states
- Operational state complexity:
  - $\text{pr}_\text{R}(L(\mathcal{A}) \cup L(\mathcal{B}))$ for two-way 2D automata:
    between $2(m+n-1)$ and $2(m+n+1)$ states
  - $\text{pr}_\text{R}(L(\mathcal{A}) \oslash L(\mathcal{B}))$ for two-way 2D automata:
    between $m+n-1$ and $2m+n$ states

Research question: Can these bounds be tightened?

(Smith and Salomaa, *JALC* article to appear)

State Complexity: Two Dimensions

- State complexity tradeoff:
  - n-state two-way 2D automaton $\rightarrow$ NFA:
    between $2n - 1$ and $2n$ states
- Operational state complexity:
  - $\text{pr}_{\text{R}}(L(\mathcal{A}) \cup L(\mathcal{B}))$ for two-way 2D automata:
    between $2(m + n - 1)$ and $2(m + n + 1)$ states
  - $\text{pr}_{\text{R}}(L(\mathcal{A}) \oslash L(\mathcal{B}))$ for two-way 2D automata:
    between $m + n - 1$ and $2m + n$ states

Research question: Can these bounds be tightened?

Research question: What bounds exist for other language operations and automaton models?

(Smith and Salomaa, *JALC* article to appear)

Table of Contents

My Research Work

Background
- Two-Dimensional Automata
- Restricted 2D Automata

Past Results and Future Questions
- Decision Problems
- Concatenation of 2D Languages
- Projections of 2D Languages
- State Complexity of 2D Automata

Other Research Problems
- Combinatorics on Words
- Bio-inspired Language Operations
- Symbolic Computation Using Automata

Other Research Problems

- I have a few other problems that I am thinking about that are not directly related to 2D automata.
- These problems relate to:
  - **Combinatorics on words**
    (Computer Science + Mathematics)
  - **Bio-inspired** language operations
    (Computer Science + Biology)
  - **Symbolic computation** using automata/languages
    (Computer Science + Software Engineering)

Combinatorics on Words

- We can use combinatorics to study patterns and sequences formed within words and languages.
- For example, we can:
  - **Enumerate all words** with a certain property
  - Determine to which **language class** words with certain properties belong
  - Connect words/languages to sequences using the **On-line Encyclopedia of Integer Sequences**
- Natural opportunities arise to write code that automatically checks conjectures, etc.

Combinatorics on Words

- We can use combinatorics to study patterns and sequences formed within words and languages.
- For example, we can:
  - **Enumerate all words** with a certain property
  - Determine to which **language class** words with certain properties belong
  - Connect words/languages to sequences using the **On-line Encyclopedia of Integer Sequences**
- Natural opportunities arise to write code that automatically checks conjectures, etc.

Research question: What are some interesting properties of 2D languages?

Combinatorics on Words

STFX UNIVERSITY

- We can use combinatorics to study patterns and sequences formed within words and languages.
- For example, we can:
  - **Enumerate all words** with a certain property
  - Determine to which **language class** words with certain properties belong
  - Connect words/languages to sequences using the **On-line Encyclopedia of Integer Sequences**
- Natural opportunities arise to write code that automatically checks conjectures, etc.

Research question: What are some interesting properties of 2D languages?

Research question: What can 2D languages tell us about 1D languages and integer sequences?

Bio-inspired Language Operations

- A **bio-inspired** language operation is an operation on formal languages that comes from a biological process or phenomenon.
  - **Overlap assembly**: uvw, where $x = uv$ and $y = vw$
  - **Splicing**: $x_1z_1z_4y_2$, where $x = x_1z_1z_2x_2$ and $y = y_1z_3z_4y_2$
  - **Site-directed insertion**: x_1uzvx_2, where $x = x_1uvx_2$ and $y = uzv$
- We can study properties like the size of an automaton recognizing these operations, decidability properties, complexity properties, etc.

(Cho, Han, Salomaa, and Smith, 2019)

Bio-inspired Language Operations

- A **bio-inspired** language operation is an operation on formal languages that comes from a biological process or phenomenon.
  - **Overlap assembly**: uvw, where $x = uv$ and $y = vw$
  - **Splicing**: $x_1 z_1 z_4 y_2$, where $x = x_1 z_1 z_2 x_2$ and $y = y_1 z_3 z_4 y_2$
  - **Site-directed insertion**: $x_1 uzv x_2$, where $x = x_1 uv x_2$ and $y = uzv$
- We can study properties like the size of an automaton recognizing these operations, decidability properties, complexity properties, etc.

> **Research question:** What other biological operations can we model with formal languages and automata?

(Cho, Han, Salomaa, and Smith, 2019)

Symbolic Computation Using Automata

- `Grail+` is a **software package** for symbolic computation, manipulating automata, languages, and other theory objects.
- It can convert finite automata to regular expressions and vice versa, minimize/determinize automata, test properties, and so on.
- Maintained at U. PEI by Prof. Cezar Câmpeanu and students.

Symbolic Computation Using Automata

- `Grail+` is a **software package** for symbolic computation, manipulating automata, languages, and other theory objects.
- It can convert finite automata to regular expressions and vice versa, minimize/determinize automata, test properties, and so on.
- Maintained at U. PEI by Prof. Cezar Câmpeanu and students.

Student research: Developing automata visualization software using `Grail+`. (Summer 2022, fully funded, won award at regional conference)

Symbolic Computation Using Automata

- `Grail+` is a **software package** for symbolic computation, manipulating automata, languages, and other theory objects.
- It can convert finite automata to regular expressions and vice versa, minimize/determinize automata, test properties, and so on.
- Maintained at U. PEI by Prof. Cezar Câmpeanu and students.

Student research: Developing automata visualization software using `Grail+`. (Summer 2022, fully funded, won award at regional conference)

Research question: How can we extend `Grail+` to use new language operations, automaton models, etc.?

References I

[1] J.-C. Birget. Partial orders on words, minimal elements of regular languages, and state complexity. *Theoret. Comput. Sci.*, 119(2):267–291, 1993.

[2] D.-J. Cho, Y.-S. Han, K. Salomaa, and T. J. Smith. Site-directed insertion: Language equations and decision problems. *Theoret. Comput. Sci.*, 798:40–51, 2019.

[3] M. Holzer and M. Kutrib. Nondeterministic descriptional complexity of regular languages. *Int. J. Found. Comput. Sci.*, 14(6):1087–1102, 2003.

[4] A. N. Maslov. Estimates of the number of states of finite automata. *Sov. Math. Dokl.*, 11(5):1373–1375, 1970.

[5] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.

[6] A. Salomaa, K. Salomaa, and T. J. Smith. Descriptional complexity of finite automata – selected highlights, 2023. arXiv:2301.03708.

[7] T. J. Smith. Two-dimensional typewriter automata. In H. Bordihn, G. Horváth, and G. Vaszil, editors, *Short Papers of NCMA 2022*, pages 38–45, Debrecen, 2022. Faculty of Informatics, University of Debrecen.

References II

[8] T. J. Smith and K. Salomaa. Recognition and complexity results for projection languages of two-dimensional automata. *J. Autom. Lang. Comb.* To appear.

[9] T. J. Smith and K. Salomaa. Recognition and complexity results for projection languages of two-dimensional automata. In G. Jirásková and G. Pighizzini, editors, *Proc. of DCFS 2020*, volume 12442 of *LNCS*, pages 206–218, Berlin Heidelberg, 2020. Springer-Verlag.

[10] T. J. Smith and K. Salomaa. Concatenation operations and restricted variants of two-dimensional automata. In T. Bureš et al., editors, *Proc. of SOFSEM 2021*, volume 12607 of *LNCS*, pages 147–158, Berlin Heidelberg, 2021. Springer-Verlag.

[11] T. J. Smith and K. Salomaa. Decision problems and projection languages for restricted variants of two-dimensional automata. *Theoret. Comput. Sci.*, 870:153–164, 2021.