Concatenation Operations and Restricted Variants of Two-Dimensional Automata

SOFSEM 2021

Taylor J. Smith

Joint work with K. Salomaa

School of Computing
Queen’s University
Kingston, Ontario, Canada

January 29, 2021
| Section                  | Subsections                                          |
|--------------------------|------------------------------------------------------|
| **Introduction**         | Two-Dimensional Automata                             |
|                          | Restricted 2D Automata                               |
|                          | Concatenation Operations                             |
| **Row/Column Concatenation** | Two-Way 2D Automata                                |
|                          | Unary Two-Way 2D Automata                            |
| **Diagonal Concatenation** | Two-Way 2D Automata                                |
|                          | Three-Way 2D Automata                                |
| **Conclusions**          |                                                     |
Table of Contents

Introduction
  Two-Dimensional Automata
  Restricted 2D Automata
  Concatenation Operations

Row/Column Concatenation
  Two-Way 2D Automata
  Unary Two-Way 2D Automata

Diagonal Concatenation
  Two-Way 2D Automata
  Three-Way 2D Automata

Conclusions
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

```
# # # ··· # #
# a_{1,1} a_{1,2} ··· a_{1,n} #
# a_{2,1} a_{2,2} ··· a_{2,n} #
: : : ··· : :
# a_{m,1} a_{m,2} ··· a_{m,n} #
# # # ··· # #
```
A two-dimensional (2D) automaton is a generalization of a one-dimensional automaton.

Two major differences:
1. Different input word
2. Different transition function

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

Deterministic
four-way
(2DFA-4W)

Nondeterministic
four-way
(2NFA-4W)
Remark
A note on notation...

2DFA-nW

- dimension of input word
- # of input head moves

Notation like “4DFA” is found in literature discussing 2DFA-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”.
  - Similar to a one-way one-dimensional automaton.
In two dimensions, we can concatenate two words \( w \) and \( v \):

- row-wise \((w \ominus v)\)
- column-wise \((w \oplus v)\)
In two dimensions, we can concatenate two words $w$ and $v$:

- **row-wise** ($w \oplus v$)
- **column-wise** ($w \odot v$)

\[
\begin{array}{cccc}
\# & \# & \cdots & \# \\
\# & w_{1,1} & \cdots & w_{1,n} \\
\vdots & \vdots & \ddots & \vdots \\
\# & w_{m,1} & \cdots & w_{m,n} \\
\# & v_{1,1} & \cdots & v_{1,n} \\
\vdots & \vdots & \ddots & \vdots \\
\# & v_{m',1} & \cdots & v_{m',n} \\
\# & \# & \cdots & \# \\
\# & \# & \cdots & \# \\
\end{array}
\]

$$w \oplus v =$$

\[
\begin{array}{cccc}
\# & \# & \cdots & \# \\
\# & w_{1,1} & \cdots & w_{1,n} & \# \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\# & w_{m,1} & \cdots & w_{m,n} & \# \\
\# & v_{1,1} & \cdots & v_{1,n} & \# \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\# & v_{m',1} & \cdots & v_{m',n} & \# \\
\# & \# & \cdots & \# \\
\# & \# & \cdots & \# \\
\end{array}
\]
In two dimensions, we can concatenate two words $w$ and $v$:

- row-wise ($w \ominus v$)
- column-wise ($w \oplus v$)

$$w \oplus v = \begin{bmatrix}
\# & \# & \# & \# & \# & \# \\
\# & w_{1,1} & \cdots & w_{1,n} & v_{1,1} & \cdots & v_{1,n'} & \#
\end{bmatrix}$$
In two dimensions, we can concatenate two words $w$ and $v$:

- row-wise ($w \ominus v$)
- column-wise ($w \oplus v$)

Anselmo et al. (2005) introduced a third operation called **diagonal concatenation** ($w \oslash v$).

\[
\begin{array}{cccccccc}
\# & \# & \# & \# & \# & \# & \# & \# \\
\# & w_{1,1} & \cdots & w_{1,n} & x_{1,1} & \cdots & x_{1,n'} & \# \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\# & y_{1,1} & \cdots & y_{1,n} & v_{1,1} & \cdots & v_{1,n'} & \# \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\# & y_{m',1} & \cdots & y_{m',n} & v_{m',1} & \cdots & v_{m',n'} & \# \\
\# & \# & \# & \# & \# & \# & \# & \# \\
\end{array}
\]

\[w \oslash v = \]
In two dimensions, we can concatenate two words \( w \) and \( v \):
- row-wise \( (w \ominus v) \)
- column-wise \( (w \oplus v) \)

Anselmo et al. (2005) introduced a third operation called diagonal concatenation \( (w \oslash v) \).

Concatenation can be extended to languages in the usual way.

\[
A \circ B = \{ a \circ b \mid a \in A \text{ and } b \in B \}
\]
## Concatenation Closure

|                  | Row ($\odot$) | Column ($\oplus$) | Diagonal ($\odot\odot$) |
|------------------|---------------|-------------------|-------------------------|
| 2DFA-4W          | X             | X                 | ?                       |
| 2NFA-4W          | X             | X                 | ?                       |
| 2DFA-3W          | X             | X                 | ?                       |
| 2NFA-3W          | ✓             | X                 | ?                       |
| 2DFA-2W          | ?             | ?                 | ?                       |
| 2NFA-2W          | ?             | ?                 | ?                       |
## Concatenation Closure

|                | Row ($\ominus$) | Column ($\oplus$) | Diagonal ($\oslash$) |
|----------------|-----------------|-------------------|----------------------|
| 2DFA-4W        | ✗               | ✗                 | ?                    |
| 2NFA-4W        | ✗               | ✗                 | ?                    |
| 2DFA-3W        | ✗               | ✗                 | ?                    |
| 2NFA-3W        | ✓               | ✗                 | ?                    |
| 2DFA-2W        | ✗ / ✓†          | ✗ / ✓†            | ?                    |
| 2NFA-2W        | ✗ / ✓†          | ✗ / ✓†            | ?                    |

†: applies to unary alphabets
## Concatenation Closure

|               | Row ($\ominus$) | Column ($\oplus$) | Diagonal ($\oslash$) |
|---------------|-----------------|-------------------|----------------------|
| 2DFA-4W       | $\times$       | $\times$          | ?                    |
| 2NFA-4W       | $\times$       | $\times$          | ?                    |
| 2DFA-3W       | $\times$       | $\times$          | $\times$             |
| 2NFA-3W       | $\checkmark$   | $\times$          | ?                    |
| 2DFA-2W       | $\times$       | $\times$          | $\times$             |
| 2NFA-2W       | $\times$ / $\checkmark$ $\dagger$ | $\times$ / $\checkmark$ $\dagger$ | $\checkmark$ |

$\dagger$: applies to unary alphabets
Table of Contents

Introduction
  Two-Dimensional Automata
  Restricted 2D Automata
  Concatenation Operations

Row/Column Concatenation
  Two-Way 2D Automata
  Unary Two-Way 2D Automata

Diagonal Concatenation
  Two-Way 2D Automata
  Three-Way 2D Automata

Conclusions
Theorem
The class 2NFA-2W is not closed under row concatenation.

Proof (not in paper)
Let $\Sigma = \{0, 1\}$, and define $L = \{w \mid \text{for all } j, w[1, j] = 0\}$. $L$ is recognized by a two-way 2D automaton.
An automaton recognizing $L \ominus L$ accepts

\[
\begin{array}{cc}
0 & 0 \\
0 & 0 \\
\end{array}
\in L \ominus L.
\]

But the accepting computation does not visit all symbols, so (for instance) the automaton may also accept

\[
\begin{array}{cc}
0 & 0 \\
1 & 0 \\
\end{array}
\notin L \ominus L.
\]
Theorem
The class 2NFA-2W is not closed under row concatenation.

▶ Adapting this to the deterministic case, we get...

Corollary
The class 2DFA-2W is not closed under row concatenation.

▶ And, following a similar proof, we get...

Corollary
The classes 2DFA-2W and 2NFA-2W are not closed under column concatenation.
Before we proceed, we must modify our model slightly.

An automaton $\mathcal{A}$ is “IBR-accepting” if, upon reading a boundary marker on the bottom/right border of the word, $\mathcal{A}$ immediately halts and accepts if it can reach $q_{\text{accept}}$ from its current state.

**Lemma**

Given a two-way 2D automaton $\mathcal{A}$, there exists an equivalent IBR-accepting two-way 2D automaton $\mathcal{A}'$. 
Unary Row Concatenation: 2NFA-2W

Theorem
The class 2NFA-2W over a unary alphabet is closed under row concatenation.

Proof Sketch
We take a case-based approach.

- Let $A$ and $B$ be IBR-accepting unary two-way 2D automata.
- Automaton $A$ recognizes language $A$ (and $B$ recognizes $B$).
- Their accepting computations are denoted by $C_A$ and $C_B$. 
Proof Sketch (cont’d)

We construct an automaton $\mathcal{M}$ to recognize $A \ominus B$. $\mathcal{M}$ nondeterministically chooses which “types” of computation correspond to $C_A$ and $C_B$, and interleaves.

“Types” of Computation

1. $C_A$ accepts at bottom, $C_B$ accepts at right
2. $C_A$ accepts at right, $C_B$ accepts at bottom
3. (a) $C_A$ accepts at bottom in column $i$, $C_B$ accepts at bottom in column $j < i$
   (b) $C_A$ accepts at bottom in column $i$, $C_B$ accepts at bottom in column $k \geq i$
4. $C_A$ and $C_B$ both accept at right
Proof Sketch (cont’d)

1. $C_A$ accepts at bottom, $C_B$ accepts at right

We divide the computation of $M$ into two phases.

First Phase

(i) Simulate downward moves of $A$ by moving input head and changing state

(ii) Simulate downward moves of $B$ by moving input head and changing state

(iii) Simulate rightward moves of $A$ and $B$ by moving input head and changing state simultaneously

After completing step (iii), return to step (i) and repeat.
Proof Sketch (cont’d)

1. $C_A$ accepts at bottom, $C_B$ accepts at right

We divide the computation of $M$ into two phases.

First Phase

(i) Simulate downward moves of $A$ by moving input head and changing state

(ii) Simulate downward moves of $B$ by moving input head and changing state

(iii) Simulate rightward moves of $A$ and $B$ by moving input head and changing state simultaneously

- In step (i), $M$ can guess nondeterministically that the input head of $A$ is at the bottom border.
- If $A$ is in an accepting state, $M$ moves to the second phase.
Proof Sketch (cont’d)

1. $C_A$ accepts at bottom, $C_B$ accepts at right

We divide the computation of $M$ into two phases.

Second Phase

- $M$ simulates the remainder of the computation of $B$.
- If $B$ is in an accepting state when the input head of $M$ reaches the right border, $M$ accepts.
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

Input word for $A$  

Input word for $B$  

➤ First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

- First phase, step: (i) (ii) (iii)
Example

1. \(C_A\) accepts at bottom, \(C_B\) accepts at right

\[\begin{array}{c}
\text{Input word for } A \\
\end{array}\]

\[\begin{array}{c}
\text{Input word for } B \\
\end{array}\]

- First phase, step: (i) (ii) (iii)
Unary Row Concatenation: 2NFA-2W

Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

Input word for $A$  

Input word for $B$  

First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

Input word for $A$

Input word for $B$

First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

First phase, step: (i) (ii) (iii)
 Unary Row Concatenation: 2NFA-2W

Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

Input word for $A$

Input word for $B$

▶ First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

Input word for $A$  
Input word for $B$

▶ First phase, step: (i) (ii) (iii)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

First phase, step: (i) (ii) (iii)
Unary Row Concatenation: 2NFA-2W

Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

\[ \text{Input word for } A \] 
\[ \text{Input word for } B \]

▶ First phase, step: (i) (ii) (iii)
Unary Row Concatenation: 2NFA-2W

Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

First phase, step: (i) (ii) (iii)

Simulation of $C_A$ accepts at (1)
Unary Row Concatenation: 2NFA-2W

Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

- First phase, step: (i) (ii) (iii)
- Simulation of $C_A$ accepts at (1)
- Computation of $M$ begins second phase at (2)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

- Second phase
- Simulation of $C_A$ accepts at (1)
- Computation of $M$ begins second phase at (2)
Unary Row Concatenation: 2NFA-2W

Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

- Second phase
- Simulation of $C_A$ accepts at (1)
- Computation of $M$ begins second phase at (2)
Example

1. $C_A$ accepts at bottom, $C_B$ accepts at right

- Second phase
- Simulation of $C_A$ accepts at (1)
- Computation of $M$ begins second phase at (2)
Example

1. \( C_A \) accepts at bottom, \( C_B \) accepts at right

- Second phase
- Simulation of \( C_A \) accepts at (1)
- Computation of \( M \) begins second phase at (2)
Theorem
The class 2NFA-2W over a unary alphabet is closed under row concatenation.

By swapping downward and rightward moves, we get . . .

Corollary
The class 2NFA-2W over a unary alphabet is closed under column concatenation.
Table of Contents

Introduction
Two-Dimensional Automata
Restricted 2D Automata
Concatenation Operations

Row/Column Concatenation
Two-Way 2D Automata
Unary Two-Way 2D Automata

Diagonal Concatenation
Two-Way 2D Automata
Three-Way 2D Automata

Conclusions
**Theorem**
The class 2NFA-2W is closed under diagonal concatenation.

**Proof (not in paper)**
Construct an automaton $C$ that simulates the computations of both original automata $A$ and $B$.
Some modifications:

- Convert first automaton $A$ to be IBR-accepting.
- Modify transition function so that $C$ accepts if and only if IBR-accepting $A$ would accept on boundary marker.
- Move input head of $C$ some **nondeterministically-selected** number of cells before simulating $B$. 
Theorem
The class 2NFA-2W is closed under diagonal concatenation.

▶ As a consequence of the construction, we get...

Theorem
The class 2DFA-2W is not closed under diagonal concatenation.
We did not have closure for the class 2DFA-2W.

We can reasonably assume that the class 2DFA-3W will also not be closed.

However, we require a different approach:

- Need to account for added direction of movement
- The input head can now read all symbols in a row
Theorem
The class 2DFA-3W is not closed under diagonal concatenation.

Key Observations

- Using a two-way 1D automaton $N$, we can simulate the computation of a three-way 2D automaton $M$ on a certain row of its input.
- The number of states of $N$ depends linearly on the number of states of $M$. 
Let $\Sigma = \{0, 1\}$, and let $\mathcal{M}$ be a deterministic three-way 2D automaton with $n$ states.

To prove our result, we need two lemmas that use $\mathcal{M}$.

First Lemma
Consider the computation of $\mathcal{M}$ on a row of all-0s. If the input head of $\mathcal{M}$ reads the first or last symbol of the row, then it moves downward to the next row at most $n + 1$ cells away from a boundary marker.

\[
\begin{array}{cccccccc}
1 & 2 & \cdots & n-1 & n & n+1 & n+2 & \cdots \\
\# & 0 & 0 & \cdots & 0 & 0 & 0 & \to \\
\downarrow & & & & & & & \\
\# & 1 & 0 & \cdots & 0 & 1 & 1 & 0
\end{array}
\]
Let $\Sigma = \{0, 1\}$, and let $\mathcal{M}$ be a deterministic three-way 2D automaton with $n$ states.

To prove our result, we need two lemmas that use $\mathcal{M}$.

**Second Lemma**

Suppose $\mathcal{M}$ enters a row at most $n + 1$ cells away from a boundary marker. Then there exists a two-way 1D automaton $\mathcal{N}$ with at most $2n + 3$ states that

1. simulates the computation of $\mathcal{M}$ on that row, and
2. accepts if and only if the input head of $\mathcal{M}$ moves downward to the next row.
Kapoutsis (2005) showed that we can take a deterministic two-way 1D automaton with $n$ states and convert it to a one-way automaton with $h(n) = n(n^n - (n - 1)^n)$ states.

We use this value to construct a “diagonal concatenation” language of words of a certain size, where each row contains certain patterns of symbols.

Proof Sketch
Using our lemmas, reach a contradiction:

- Start with a three-way 2D automaton $\mathcal{C}$ with $n$ states
- Convert to a two-way 1D automaton $\mathcal{D}$ with $2n + 3$ states
- Convert to a one-way 1D automaton $\mathcal{D}'$ with $h(2n + 3)$ states
- Problem: $\mathcal{D}'$ becomes incapable of recognizing rows of words accepted by $\mathcal{C}$!
Table of Contents

Introduction
  Two-Dimensional Automata
  Restricted 2D Automata
  Concatenation Operations

Row/Column Concatenation
  Two-Way 2D Automata
  Unary Two-Way 2D Automata

Diagonal Concatenation
  Two-Way 2D Automata
  Three-Way 2D Automata

Conclusions
Conclusions

- 2D automata can be restricted to move in fewer than four directions.
- Depending on the model, two concatenated words from some class of languages may or may not belong to that same class.
- Neither row nor column concatenation is closed for two-way 2D automata...
  - ...except in the unary nondeterministic case.
- Diagonal concatenation is closed for nondeterministic two-way 2D automata...
  - ...but not in the deterministic two-way or deterministic three-way cases.
Future Work

- What kind of closure results can we get for other models in the unary case?
- Do we have closure for diagonal concatenation on four-way 2D automata?
- Do we have closure for diagonal concatenation on nondeterministic three-way 2D automata?
  - Conjecture: no, but we require essentially a different approach.
[1] M. Anselmo, D. Giammarresi, and M. Madonia. New operations and regular expressions for two-dimensional languages over one-letter alphabet. *Theoret. Comput. Sci.*, 340(2):408–431, 2005.

[2] C. Kapoutsis. Removing bidirectionality from nondeterministic finite automata. In J. Jędrzejowicz and A. Szepietowski, editors, *Proc. of MFCS 2005*, volume 3618 of *LNCS*, pages 544–555, Berlin Heidelberg, 2005. Springer-Verlag.

[3] 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.