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

Table of Contents

Table of Contents

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**

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

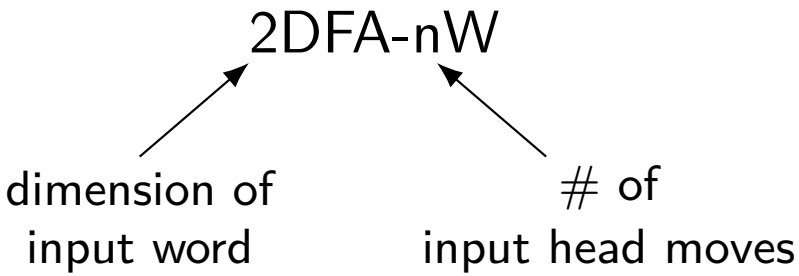
Deterministic four-way (2DFA-4W)

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

Nondeterministic four-way (2NFA-4W)

Remark

A note on notation. . .



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.

Concatenation Operations

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

Concatenation Operations

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

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

Concatenation Operations

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

$$w \oplus v = \begin{array}{cccccccc} \# & \# & & \# & \# & & \# & \# \\ \# & w_{1,1} & \cdots & w_{1,n} & v_{1,1} & \cdots & v_{1,n'} & \# \\ & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \\ \# & w_{m,1} & \cdots & w_{m,n} & v_{m,1} & \cdots & v_{m,n'} & \# \\ \# & \# & & \# & \# & & \# & \# \end{array}$$

Concatenation Operations

- 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** $(w \oslash v)$.

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

Concatenation Operations

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

†: applies to unary alphabets

Table of Contents

Row Concatenation: 2NFA-2W

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

$$\begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \in L \ominus L.$$

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

$$\begin{matrix} 0 & 0 \\ 1 & 0 \end{matrix} \notin L \ominus L.$$

Row/Col Concatenation: 2DFA-2W/2NFA-2W

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.

Unary Row Concatenation: Preliminaries

- 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_{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 $\mathcal{A}$ and $\mathcal{B}$ be IBR-accepting unary two-way 2D automata.
- Automaton $\mathcal{A}$ recognizes language A (and $\mathcal{B}$ recognizes B).
- Their accepting computations are denoted by $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$.

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 $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$, and interleaves.

"Types" of Computation

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right
2. $C_{\mathcal{A}}$ accepts at right, $C_{\mathcal{B}}$ accepts at bottom
3. (a) $C_{\mathcal{A}}$ accepts at bottom in column i, $C_{\mathcal{B}}$ accepts at bottom in column $j < i$
   (b) $C_{\mathcal{A}}$ accepts at bottom in column i, $C_{\mathcal{B}}$ accepts at bottom in column $k \geq i$
4. $C_{\mathcal{A}}$ and $C_{\mathcal{B}}$ both accept at right

Proof Sketch (cont'd)

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right

We divide the computation of $\mathcal{M}$ into two phases.

First Phase

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

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

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

After completing step (iii), return to step (i) and repeat.

Proof Sketch (cont'd)

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right

We divide the computation of $\mathcal{M}$ into two phases.

First Phase

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

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

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

- In step (i), $\mathcal{M}$ can guess nondeterministically that the input head of $\mathcal{A}$ is at the bottom border.
- If $\mathcal{A}$ is in an accepting state, $\mathcal{M}$ moves to the second phase.

Proof Sketch (cont'd)

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right

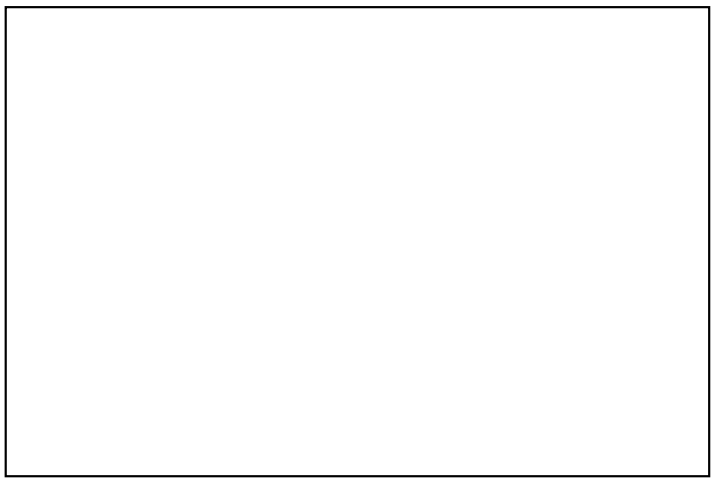
We divide the computation of $\mathcal{M}$ into two phases.

Second Phase

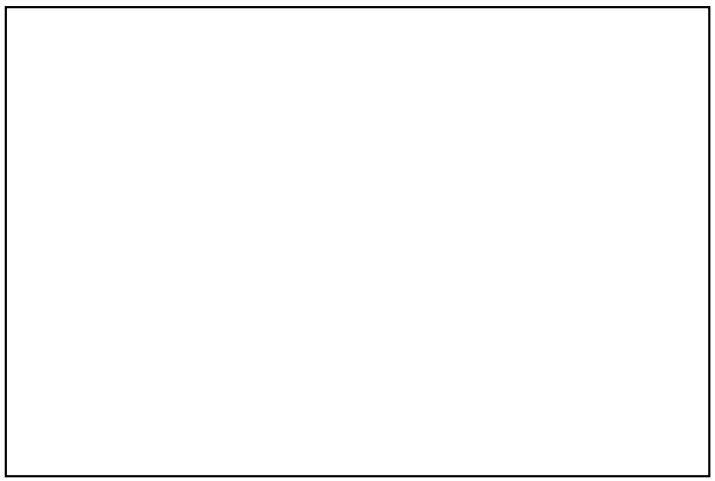
- $\mathcal{M}$ simulates the remainder of the computation of $\mathcal{B}$.
- If $\mathcal{B}$ is in an accepting state when the input head of $\mathcal{M}$ reaches the right border, $\mathcal{M}$ accepts.

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



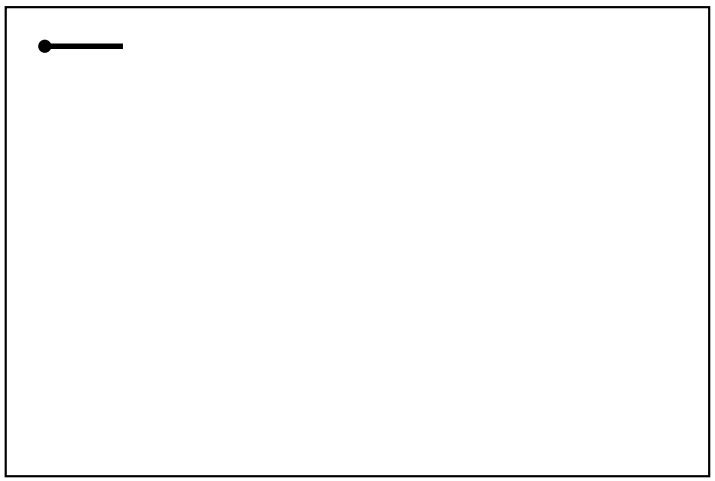
Input word for $\mathcal{B}$

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

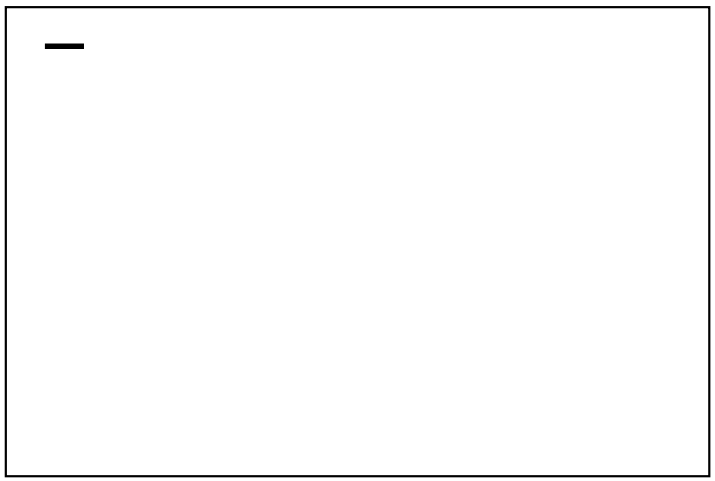
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$

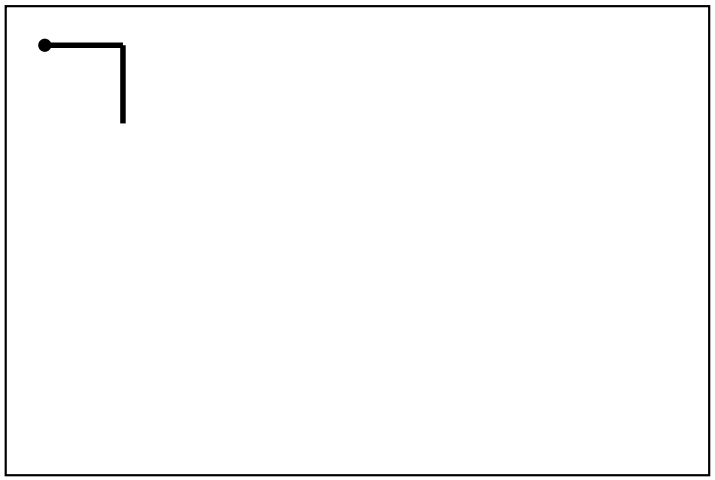


Input word for $\mathcal{B}$

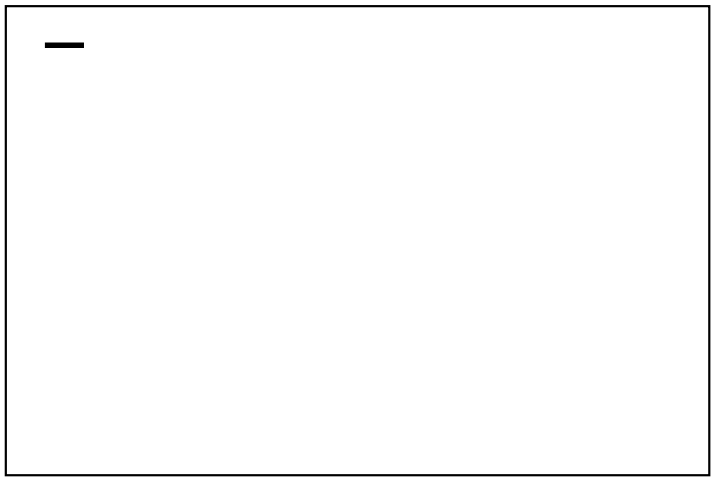
- First phase, step: (i) (ii) **(iii)**

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$

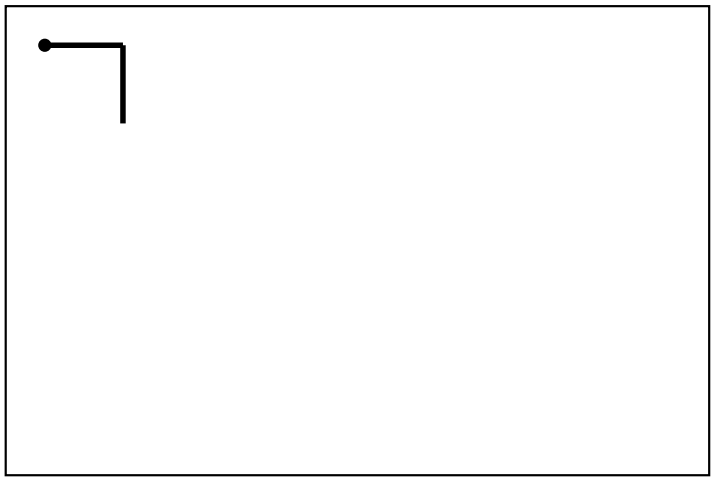


Input word for $\mathcal{B}$

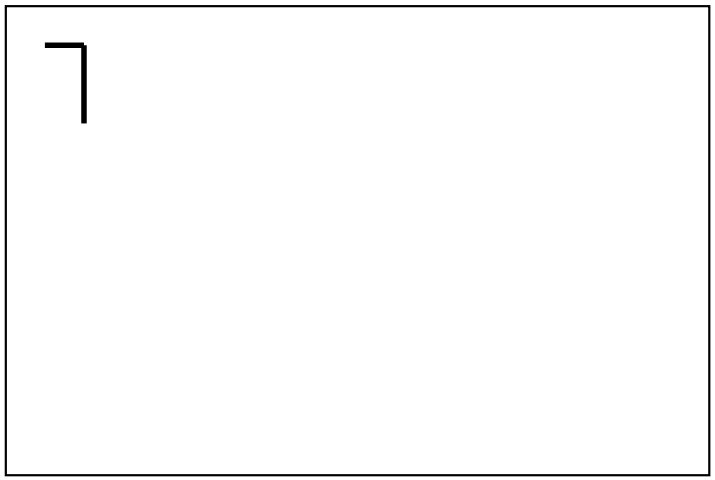
- First phase, step: **(i)** (ii) (iii)

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



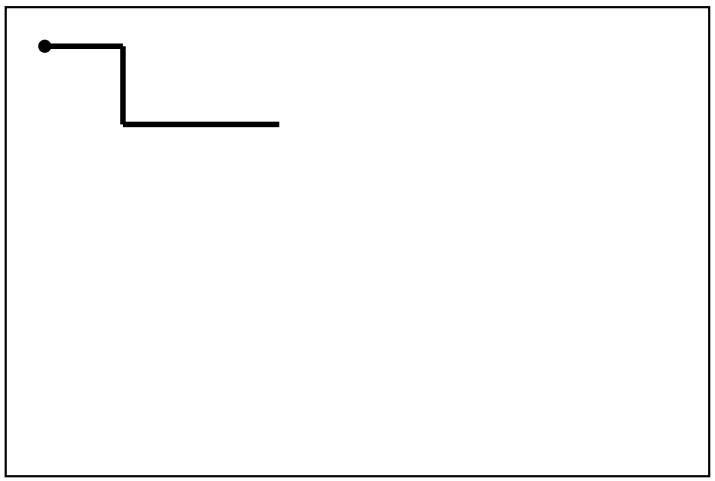
Input word for $\mathcal{B}$

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

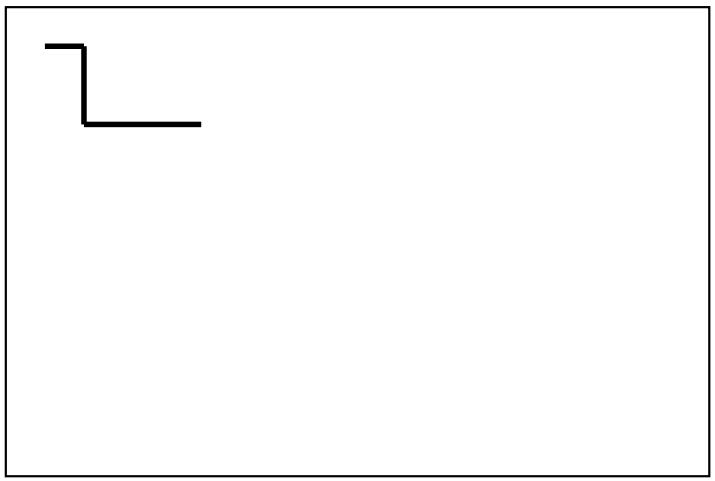
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



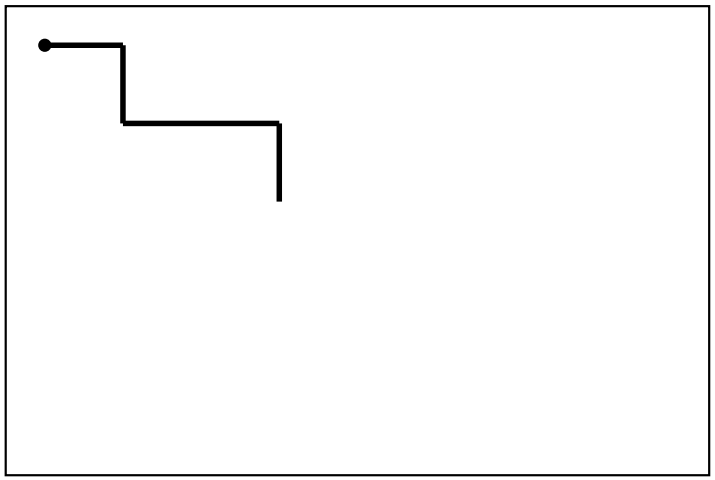
Input word for $\mathcal{B}$

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

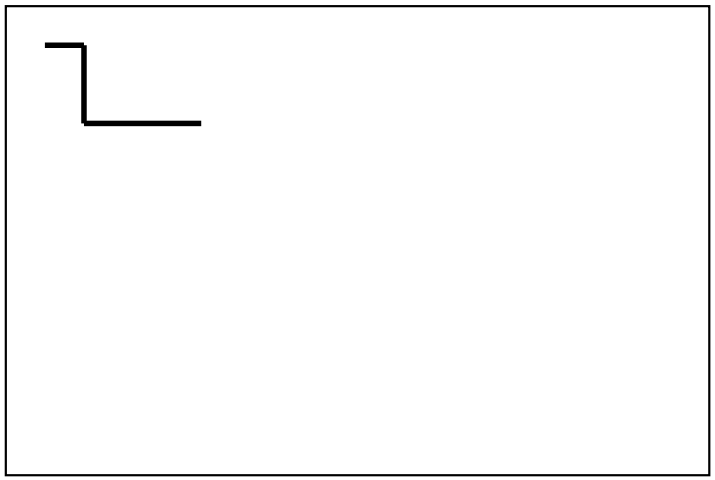
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



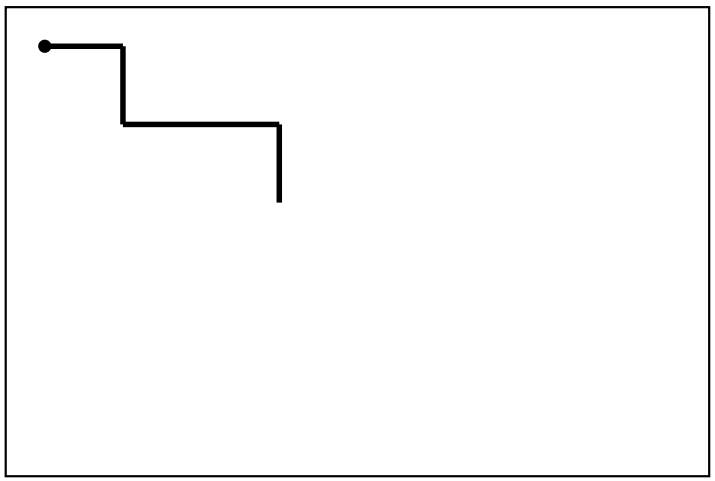
Input word for $\mathcal{B}$

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

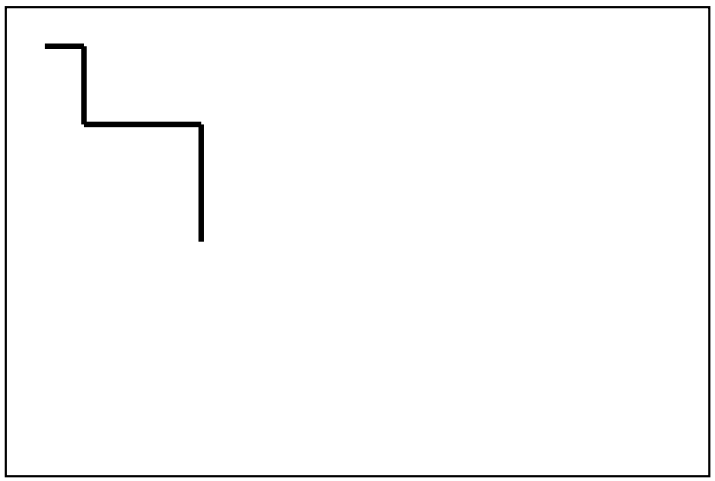
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



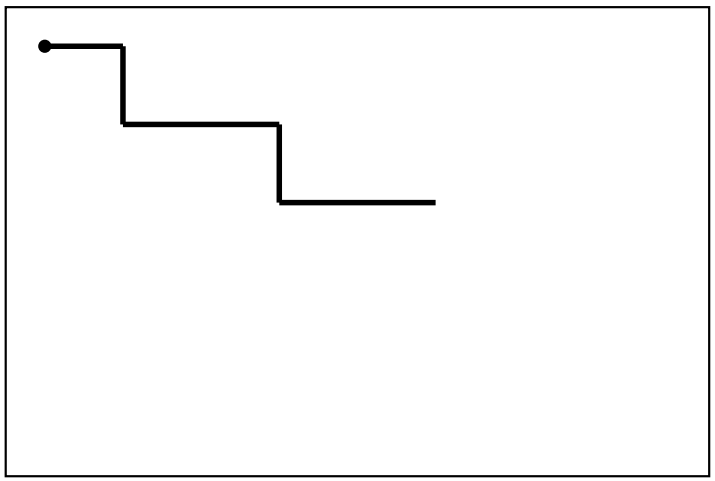
Input word for $\mathcal{B}$

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

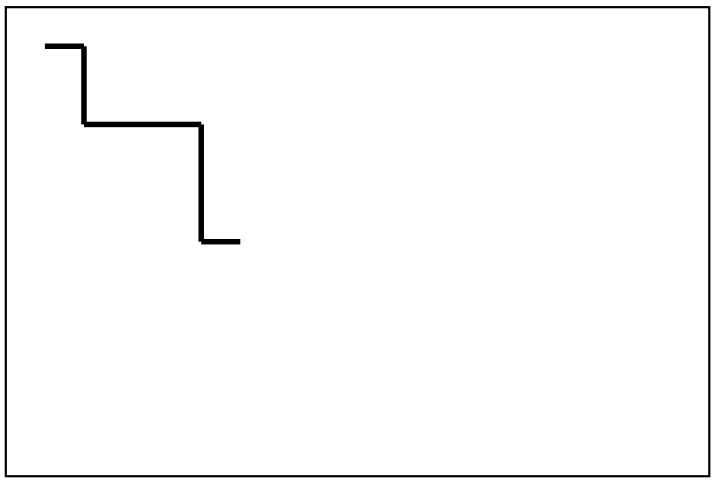
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



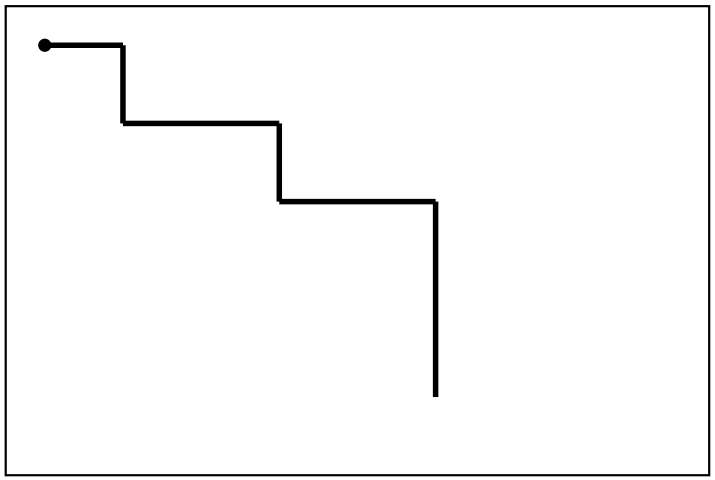
Input word for $\mathcal{B}$

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

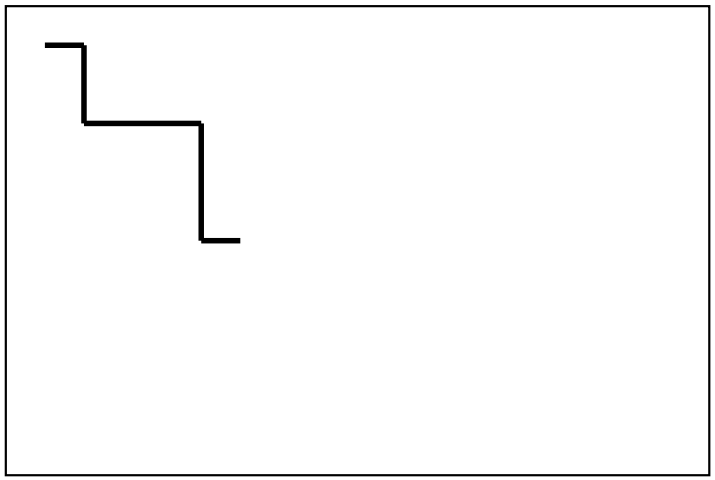
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



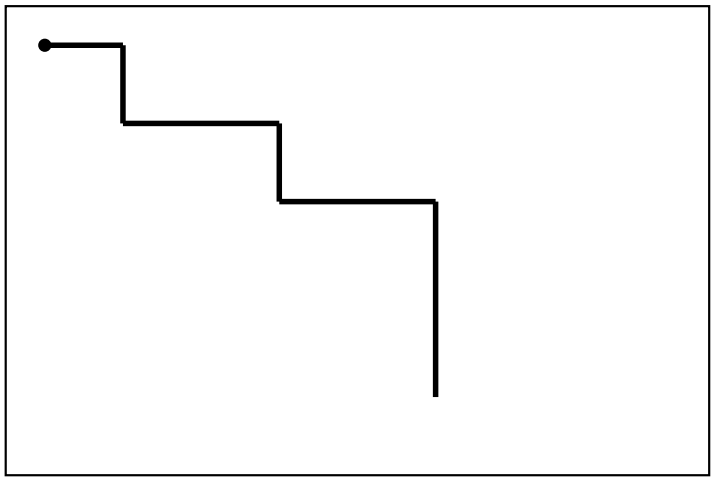
Input word for $\mathcal{B}$

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

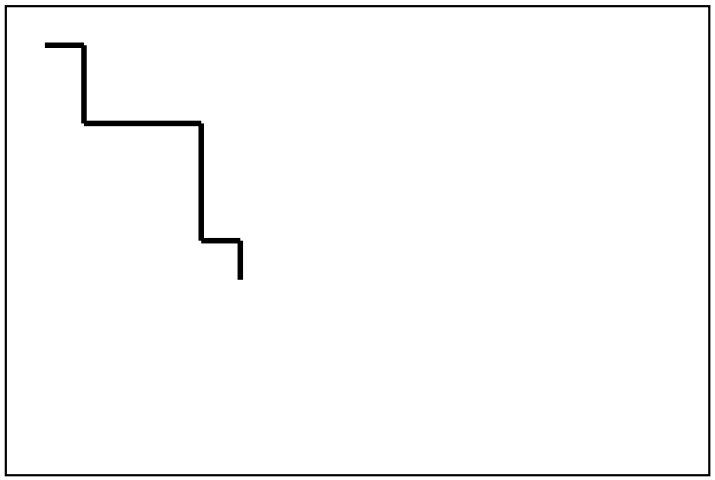
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



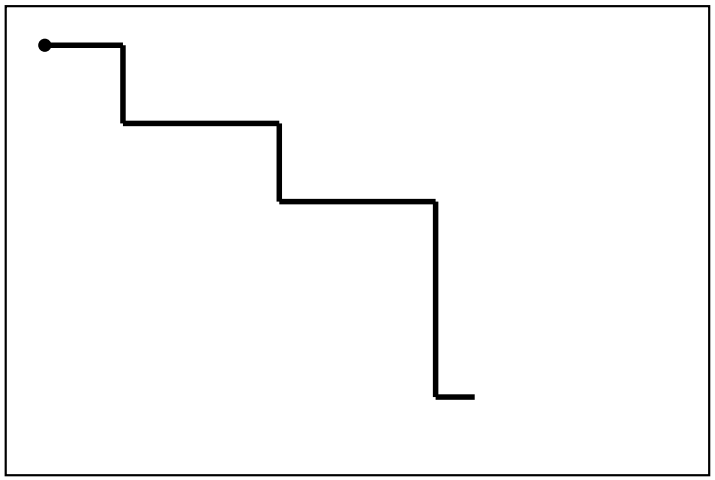
Input word for $\mathcal{B}$

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

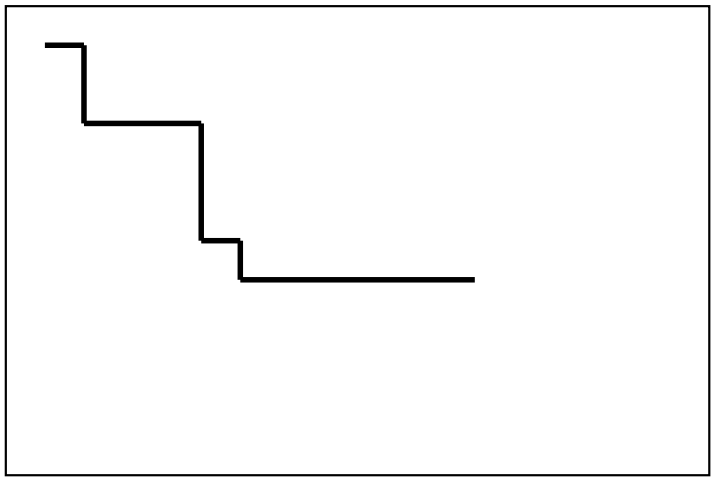
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$

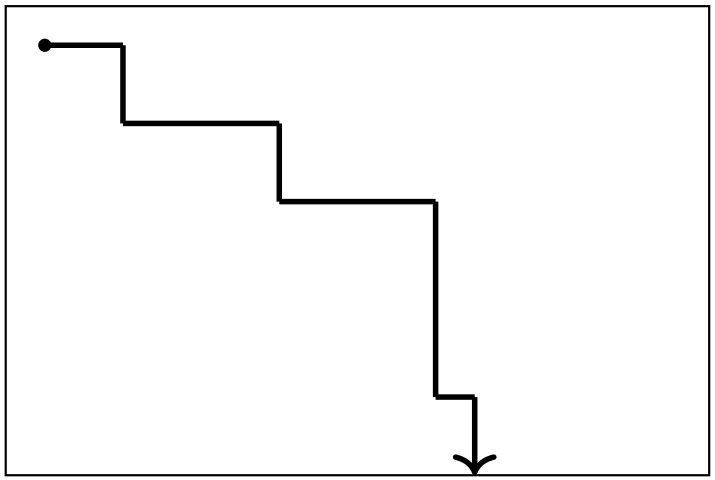


Input word for $\mathcal{B}$

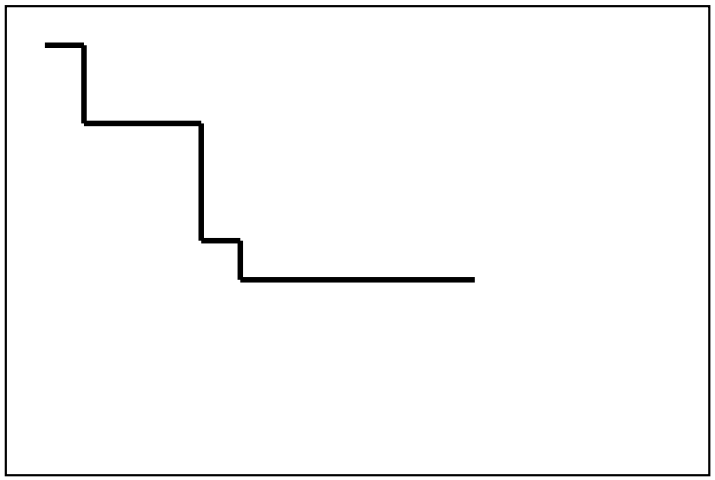
- First phase, step: (i) (ii) **(iii)**

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



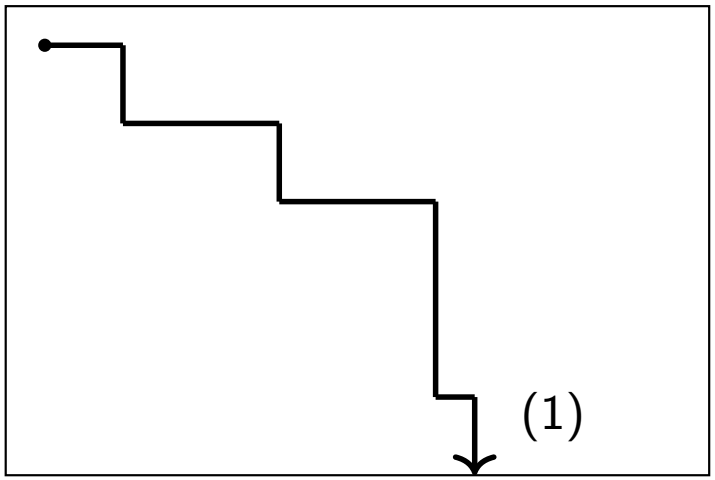
Input word for $\mathcal{B}$

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

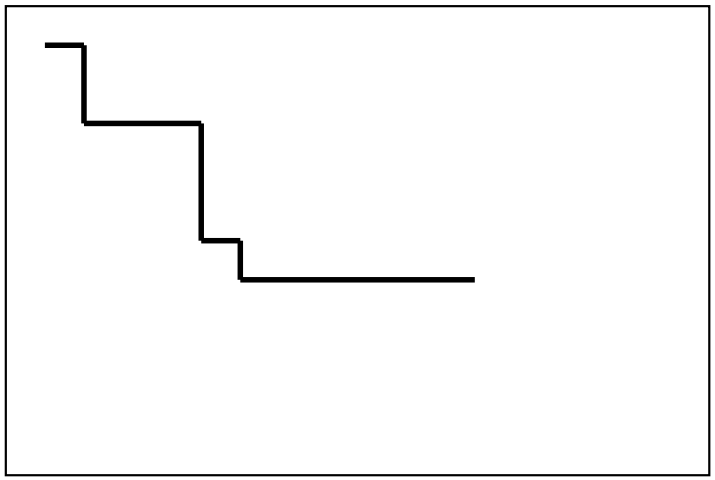
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



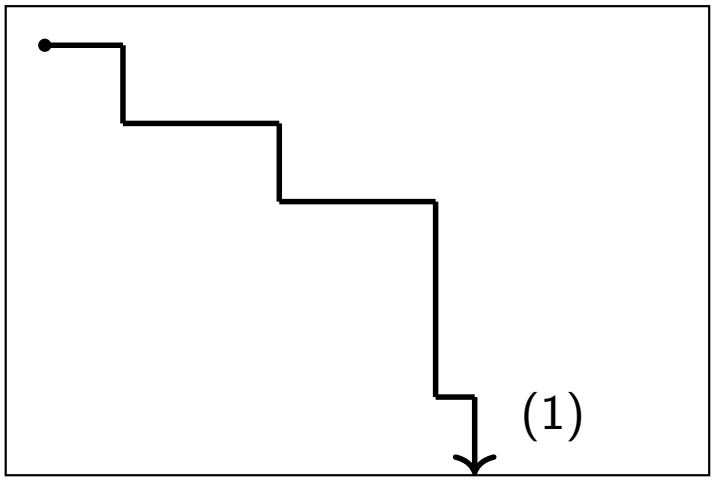
Input word for $\mathcal{B}$

- First phase, step: **(i)** (ii) (iii)
- Simulation of $C_{\mathcal{A}}$ accepts at (1)

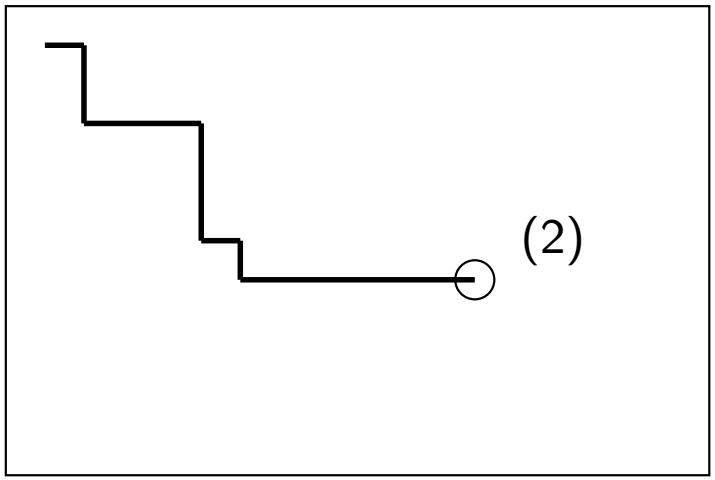
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$

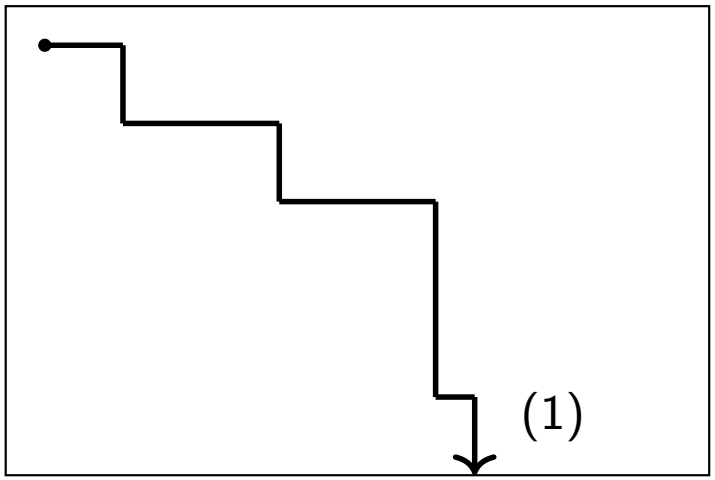


Input word for $\mathcal{B}$

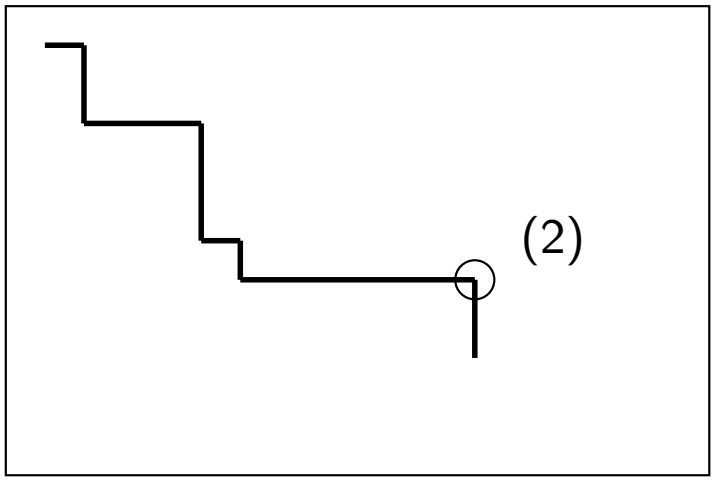
- First phase, step: **(i)** (ii) (iii)
- Simulation of $C_{\mathcal{A}}$ accepts at (1)
- Computation of $\mathcal{M}$ begins second phase at (2)

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$

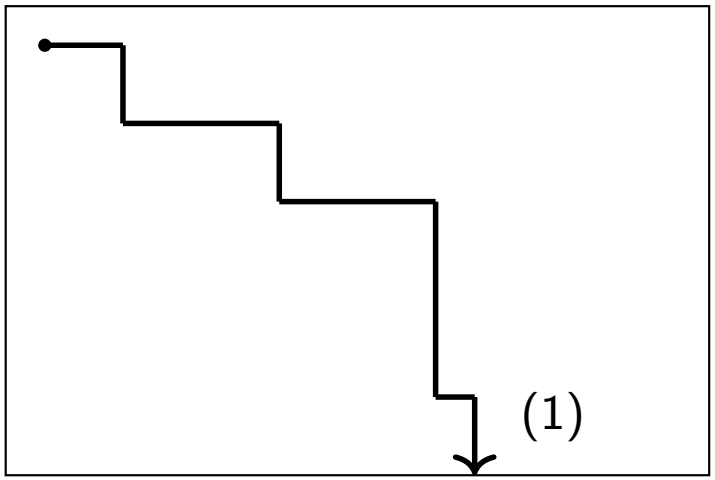


Input word for $\mathcal{B}$

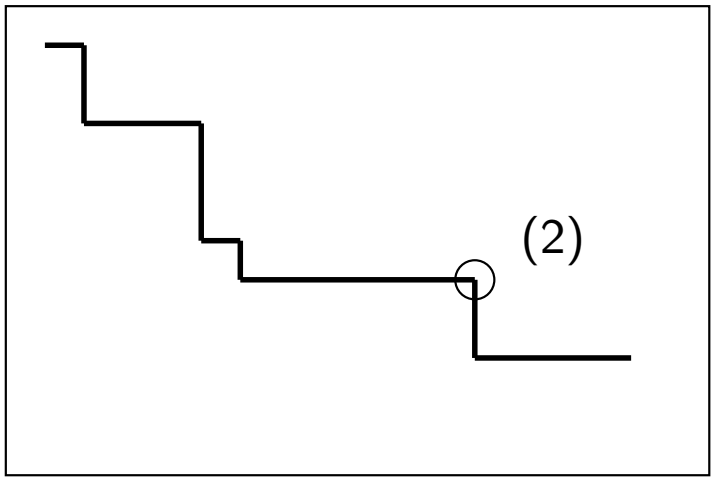
- Second phase
- Simulation of $C_{\mathcal{A}}$ accepts at (1)
- Computation of $\mathcal{M}$ begins second phase at (2)

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



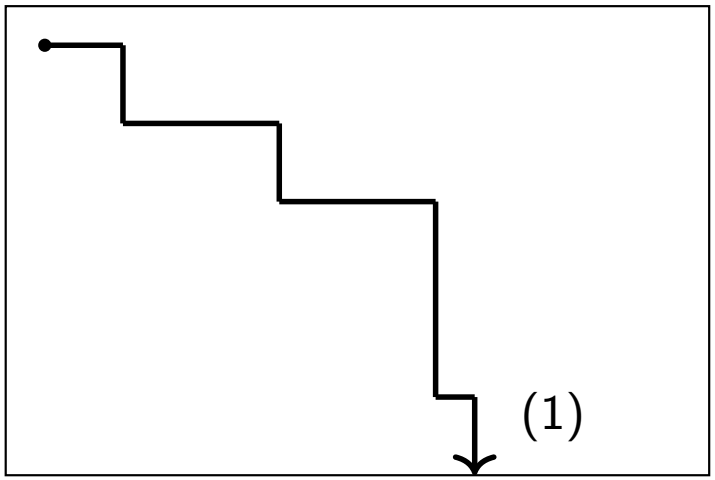
Input word for $\mathcal{B}$

- Second phase
- Simulation of $C_{\mathcal{A}}$ accepts at (1)
- Computation of $\mathcal{M}$ begins second phase at (2)

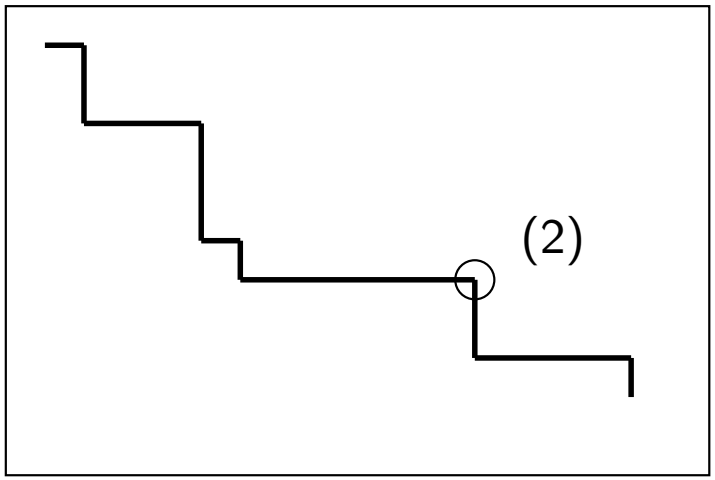
Unary Row Concatenation: 2NFA-2W

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$

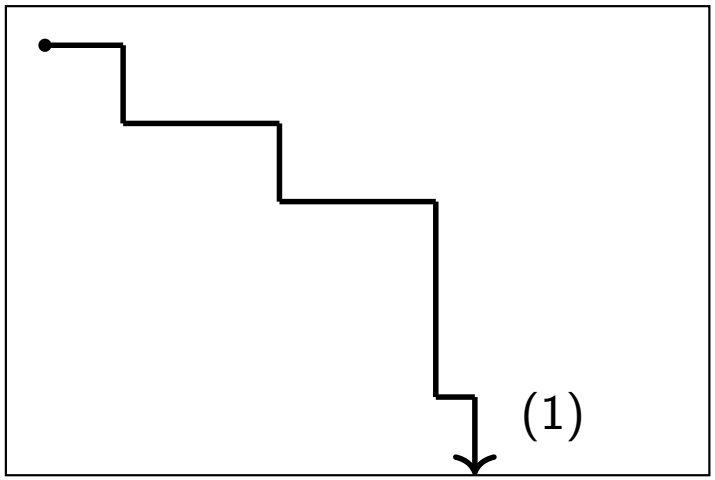


Input word for $\mathcal{B}$

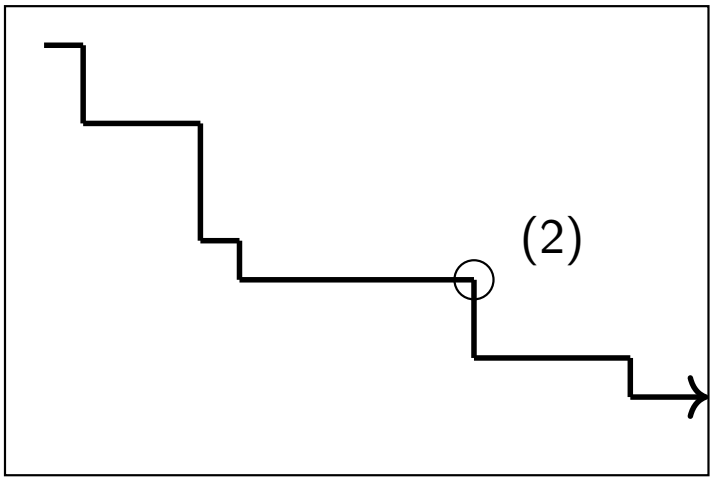
- Second phase
- Simulation of $C_{\mathcal{A}}$ accepts at (1)
- Computation of $\mathcal{M}$ begins second phase at (2)

Example

1. $C_{\mathcal{A}}$ accepts at bottom, $C_{\mathcal{B}}$ accepts at right



Input word for $\mathcal{A}$



Input word for $\mathcal{B}$

- Second phase
- Simulation of $C_{\mathcal{A}}$ accepts at (1)
- Computation of $\mathcal{M}$ begins second phase at (2)

Unary Row/Col Concatenation: 2NFA-2W

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

Diagonal Concatenation: 2NFA-2W

Theorem

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

Proof (not in paper)

Construct an automaton $\mathcal{C}$ that simulates the computations of both original automata $\mathcal{A}$ and $\mathcal{B}$.
Some modifications:

- Convert first automaton $\mathcal{A}$ to be IBR-accepting.
- Modify transition function so that $\mathcal{C}$ accepts if and only if IBR-accepting $\mathcal{A}$ would accept on boundary marker.
- Move input head of $\mathcal{C}$ some **nondeterministically-selected** number of cells before simulating $\mathcal{B}$.

Diagonal Concatenation: 2DFA-2W/2NFA-2W

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.

Diagonal Concatenation: 2DFA-3W

- 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

Diagonal Concatenation: 2DFA-3W

Theorem

The class 2DFA-3W is not closed under diagonal concatenation.

Key Observations

- Using a two-way 1D automaton $\mathcal{N}$, we can simulate the computation of a three-way 2D automaton $\mathcal{M}$ on a certain row of its input.
- The number of states of $\mathcal{N}$ depends linearly on the number of states of $\mathcal{M}$.

Diagonal Concatenation: 2DFA-3W

- 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.

	1	2	$\cdots$	$n-1$	n	$n+1$	$n+2$	$\cdots$
#	0	0	$\cdots$	0	⓪	0	⓪	$\rightarrow$
					$\downarrow$			
#	1	0	$\cdots$	0	1	1	0	

Diagonal Concatenation: 2DFA-3W

- 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.

Diagonal Concatenation: 2DFA-3W

- 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

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.

References

[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.