Decision Problems for Restricted Variants of Two-Dimensional Automata
CIAA 2019

Taylor J. Smith
Joint work with K. Salomaa
School of Computing
Queen’s University
Kingston, Ontario, Canada

July 22, 2019
Table of Contents

Introduction
   Two-Dimensional Automata
   Restricted 2D Automata
   Decision Problems

Language Emptiness
   Unary Three-Way Nondeterministic
   Two-Way Nondeterministic

Language Equivalence
   Two-Way Deterministic

Row/Column Projection

Conclusions
## Table of Contents

**Introduction**
- Two-Dimensional Automata
- Restricted 2D Automata
- Decision Problems

**Language Emptiness**
- Unary Three-Way Nondeterministic
- Two-Way Nondeterministic

**Language Equivalence**
- Two-Way Deterministic

**Row/Column Projection**

**Conclusions**
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 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.
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.
## Decision Problems

<table>
<thead>
<tr>
<th></th>
<th>2DFA-4W</th>
<th>2NFA-4W</th>
<th>2DFA-3W</th>
<th>2NFA-3W</th>
<th>2DFA-2W</th>
<th>2NFA-2W</th>
</tr>
</thead>
<tbody>
<tr>
<td>membership</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>emptiness</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>universality</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td>equivalence</td>
<td>✗</td>
<td>✗</td>
<td>?</td>
<td>✗</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
# Decision Problems

<table>
<thead>
<tr>
<th></th>
<th>2DFA-4W</th>
<th>2NFA-4W</th>
<th>2DFA-3W</th>
<th>2NFA-3W</th>
<th>2DFA-2W</th>
<th>2NFA-2W</th>
</tr>
</thead>
<tbody>
<tr>
<td>membership</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>emptiness</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✔️†</td>
<td>✔️*</td>
<td>✔️*</td>
</tr>
<tr>
<td>universality</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>?</td>
</tr>
<tr>
<td>equivalence</td>
<td>✗</td>
<td>✗</td>
<td>?</td>
<td>✗</td>
<td>✔️</td>
<td>?</td>
</tr>
</tbody>
</table>

- 0: new decidability result
- *: new complexity bound over general alphabets
- †: new complexity bound over unary alphabets
### Decision Problems

<table>
<thead>
<tr>
<th></th>
<th>2DFA-4W</th>
<th>2NFA-4W</th>
<th>2DFA-3W</th>
<th>2NFA-3W</th>
<th>2DFA-2W</th>
<th>2NFA-2W</th>
</tr>
</thead>
<tbody>
<tr>
<td>membership</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>emptiness</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓†</td>
<td>✓*</td>
<td>✓*</td>
</tr>
<tr>
<td>universality</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>equivalence</td>
<td>✗</td>
<td>✗</td>
<td>?</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

- ○: new decidability result
- ✓: new complexity bound over general alphabets
- †: new complexity bound over unary alphabets
- ○: new decidability result, post-publication
Table of Contents

Introduction
  Two-Dimensional Automata
  Restricted 2D Automata
  Decision Problems

Language Emptiness
  Unary Three-Way Nondeterministic
  Two-Way Nondeterministic

Language Equivalence
  Two-Way Deterministic

Row/Column Projection

Conclusions
Language Emptiness

- Language emptiness asks: given an automaton $\mathcal{A}$, is $L(\mathcal{A}) = \emptyset$?
- Known to be decidable for three-way 2D automata.
  - Also PSPACE-hard for general alphabets.
- Easily seen to be decidable for two-way 2D automata.
  - Simply guess an accepting computation.
The emptiness problem for two-way 1D automata is NP-complete.
▶ See Galil (1976).
▶ A unary two-way 1D automaton is a special case of a unary three-way 2D automaton.
▶ The 2D problem is at least NP-hard.

**Theorem**
The emptiness problem for unary 2NFA-3W is NP-complete.

**Proof Sketch**
Given a unary 2NFA-3W automaton $A$, replace all downward moves with “stay-in-place” moves. Call the new automaton $A'$. Evidently, $L(A) = \emptyset$ iff $L(A') = \emptyset$. This is equivalent to deciding emptiness for a unary two-way 1D automaton.
Language Emptiness: 2NFA-2W

- The proof of decidability of emptiness for two-way nondeterministic 2D automata is also a proof that the problem is in NP.
- It is possible to improve this bound.

Theorem

The emptiness problem for 2NFA-2W is in P.

Proof Sketch

Via a reachability procedure. If $q_{\text{accept}}$ is reachable, then the language is not empty. Given a 2NFA-2W automaton $\mathcal{A}$ with $n$ states, we can check reachability in polynomial time without nondeterminism.
The proof of decidability of emptiness for two-way nondeterministic 2D automata is also a proof that the problem is in NP.

It is possible to improve this bound.

**Corollary**

The emptiness problem for 2DFA-2W is in P.
# Table of Contents

Introduction
- Two-Dimensional Automata
- Restricted 2D Automata
- Decision Problems

Language Emptiness
- Unary Three-Way Nondeterministic
- Two-Way Nondeterministic

Language Equivalence
- Two-Way Deterministic

Row/Column Projection

Conclusions
Language Equivalence

- Language equivalence asks: given two automata $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$?
- Known to be undecidable for four-way 2D automata.
  - See Blum/Hewitt (1967).
- Known to be undecidable for three-way nondeterministic 2D automata.
  - See Inoue/Takanami (1980).
- Unknown if decidable for three-way deterministic 2D automata.
What about two-way 2D automata?
- Seems like it should be decidable.
- The argument is not very straightforward, though.

**Theorem**
The equivalence problem for 2DFA-2W is decidable.

**Proof Sketch**
Via a technical lemma.
Lemma

Let $A$ and $B$ be 2DFA-2W with $m$ and $n$ states, respectively. Denote $z = m \cdot n \cdot |\Sigma|^2 + 1$ and $f(z) = z^2 \cdot (z^2 + z - 1)$. If $L(A) - L(B) \neq \emptyset$, then $L(A) - L(B)$ contains a 2D word with at most $f(z)$ rows and $f(z)$ columns.

- In what follows, assume we have a word $W \in L(A) - L(B)$.
- Intuitively speaking:
  - Suppose $A$ has an accepting computation $C_A$ on $W$.
  - Suppose $B$ has a rejecting computation $C_B$ on $W$.
  - The lemma uses a pumping property to reduce the dimension of $W$ by removing shared states of $C_A$ and $C_B$.
  - This gives a witness word of bounded dimension separating $L(A)$ and $L(B)$.
Proof Sketch

**Case 1:** \( C_A \) and \( C_B \) share at least \( z \) positions. 
At least two of these shared positions must have been reached from the same states of \( A \) and \( B \) on the same symbol. 
We can reduce dimension by identifying/removing these positions.
Proof Sketch (cont’d)

**Case 2:** $C_A$ and $C_B$ share fewer than $z$ positions.
There exists some subword $Z$ consisting of $z \cdot (z^2 + z - 1)$
consecutive complete columns of the input word where $C_A$ and $C_B$
do not intersect.
Proof Sketch (cont’d)

**Case 2:** $C_A$ and $C_B$ share fewer than $z$ positions.
There exists some subword $Z$ consisting of $z \cdot (z^2 + z - 1)$
consecutive complete columns of the input word where $C_A$ and $C_B$
do not intersect.

**Case 2(a):** One or both of $C_A$ and $C_B$ do not enter $Z$.
We can reduce dimension without affecting $C_A$ or $C_B$. 
Proof Sketch (cont’d)

**Case 2:** $C_A$ and $C_B$ share fewer than $z$ positions.
There exists some subword $Z$ consisting of $z \cdot (z^2 + z - 1)$ consecutive complete columns of the input word where $C_A$ and $C_B$ do not intersect.

**Case 2(b):** Both $C_A$ and $C_B$ enter $Z$ without visiting all columns. Assume that $C_A$ is above $C_B$ and that $C_A$ continues to the last row of $Z$.
We can reduce the columns without affecting $C_A$ and $C_B$. 
Proof Sketch (cont’d)

**Case 2:** \( C_A \) and \( C_B \) share fewer than \( z \) positions.
There exists some subword \( Z \) consisting of \( z \cdot (z^2 + z - 1) \) consecutive complete columns of the input word where \( C_A \) and \( C_B \) do not intersect.

**Case 2(c):** \( C_A \) has a vertical drop of \( z \) steps within \( Z \), or \( C_A \) finishes computation at least \( z \) positions higher than \( C_B \).
We can reduce the rows without affecting \( C_A \) and \( C_B \).
Proof Sketch (cont’d)

At this point, we know that:

▶ $C_A$ is above $C_B$ within $Z$; and
▶ the vertical distance between each computation at the end is at most $z$.

Let $\max_Z$ denote the maximal vertical difference of $C_A$ and $C_B$ at any fixed column in $Z$. 
Proof Sketch (cont’d)

Case 2(c)(i): Assume $\max_{Z} z \geq z^2 + z$.
Suppose the leftmost column of maximal vertical distance is $k$.
There must exist $z$ columns between $k$ and the last column of $Z$ where vertical distance monotonically decreases or stays the same.
In two of these columns, the states of $C_A$ and $C_B$ and corresponding alphabet symbols coincide.
We can reduce the columns without affecting $C_A$ or $C_B$. 

\[
\begin{align*}
\text{max}_Z z & \geq z^2 + z \\
k & \quad \quad \quad \\
C_A & \quad \quad \quad \\
C_B & \quad \quad \quad 
\end{align*}
\]
Proof Sketch (cont’d)

Case 2(c)(ii): Assume $\max Z \leq z^2 + z - 1$.
Since $Z$ consists of $z \cdot (z^2 + z - 1)$ columns, there must exist $z$ columns within $Z$ where vertical distance stays the same.
In two of these columns, the states of $C_A$ and $C_B$ and corresponding alphabet symbols coincide.
We can reduce the columns without affecting $C_A$ or $C_B$. 
The previous cases give a method to reduce the columns of the input word.

The method to reduce rows is analogous.

Altogether, the lemma gives a brute-force algorithm to decide equivalence by focusing only on input words up to a given dimension.
Benefits of Approach

▶ The same argument also works for the inclusion problem.
▶ The inclusion problem is therefore decidable for 2DFA-2W.

Detriments of Approach

▶ The algorithm given by the technical lemma is inefficient.
▶ The same argument cannot be used for 2DFA-3W.
Table of Contents

Introduction
  Two-Dimensional Automata
  Restricted 2D Automata
  Decision Problems

Language Emptiness
  Unary Three-Way Nondeterministic
  Two-Way Nondeterministic

Language Equivalence
  Two-Way Deterministic

Row/Column Projection

Conclusions
Row/column projections are operations on 2D words.

The row (resp., column) projection of a 2D language \( L \) is the 1D language consisting of all first rows (resp., columns) of 2D words in \( L \).

\[
\begin{array}{cccccc}
\# & \# & \# & \# & \# & \#
\\
\# & 1 & 0 & 1 & \# &
\\
\# & 0 & 1 & 0 & \# &
\\
\# & 1 & 1 & 1 & \# &
\\
\# & 0 & 0 & 0 & \# &
\\
\# & \# & \# & \# & \#
\end{array}
\]

\[\downarrow\text{ (row proj.)}\]

\[
\begin{array}{cccc}
\# & 1 & 0 & 1 & \#
\end{array}
\]
Row/Column Projection

- Row/column projections over four-way 2D automata do not always produce regular languages.
  - This model can recognize words of dimension $2^n \times 2^n$.
  - Doesn’t work even for a unary alphabet.
- What about three-way/two-way 2D automata?
  - These models cannot recognize exponential-dimension words.
Theorem
The row projection language $L(A)$ of a unary 2NFA-3W $A$ is regular.

Proof Sketch
Replace downward moves with “stay-in-place” moves, as mentioned earlier.
Theorem
The column projection language $L(\mathcal{A})$ of a unary 2DFA-3W $\mathcal{A}$ is not always regular.

Proof Sketch
Construct a unary 2DFA-3W whose language is equal to

$$L_{\text{composite}} = \{a^m | m > 1 \text{ and } m \text{ is not prime}\}.$$
Theorem
The row projection language $L(A)$ of a general-alphabet 2NFA-2W $A$ is regular.

Proof Sketch
Construct a one-way nondeterministic 1D automaton $B$ with “stay-in-place” moves to simulate the computation of $A$. $B$ nondeterministically guesses symbols that $A$ reads when it moves downward and rightward. $B$ measures the number of rightward moves it made and the length of remaining input, and checks if there exists a matching accepting computation of $A$. 
Corollary

The column projection language $L(\mathcal{A})$ of a general-alphabet 2NFA-2W $\mathcal{A}$ is regular.
Row/Column Projection

<table>
<thead>
<tr>
<th></th>
<th>2DFA-4W</th>
<th>2NFA-4W</th>
<th>2DFA-3W</th>
<th>2NFA-3W</th>
<th>2DFA-2W</th>
<th>2NFA-2W</th>
</tr>
</thead>
<tbody>
<tr>
<td>row projection</td>
<td>✗</td>
<td>✗</td>
<td>✓*</td>
<td>✓*</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>column projection</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓: always regular
✗: sometimes non-regular
*: unary alphabets only
# Table of Contents

**Introduction**  
- Two-Dimensional Automata  
- Restricted 2D Automata  
- Decision Problems

**Language Emptiness**  
- Unary Three-Way Nondeterministic  
- Two-Way Nondeterministic

**Language Equivalence**  
- Two-Way Deterministic

**Row/Column Projection**

**Conclusions**
Conclusions

- 2D automata can be restricted to move in fewer than four directions.
- This restriction brings about interesting decidability properties.
- Language emptiness is NP-complete for unary 2NFA-3W.
- Language emptiness is in P for 2DFA-2W/2NFA-2W.
- Language equivalence is decidable for 2DFA-2W.
- Row projections are always regular over unary three-way 2D automata.
- Row/column projections are always regular over two-way 2D automata.
Future Work

▶ For general-alphabet three-way 2D automata, is the emptiness problem in PSPACE?
▶ Does there exist an efficient algorithm to decide equivalence for 2DFA-2W?
▶ Is the equivalence problem decidable for 2DFA-3W?
▶ What else can we study about restricted 2D automaton models?


